НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____ Олександр Коваль
«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Геометричне моделювання в інформаційних системах»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Система аналітики контролю та захисту»

Виконав (-ла):
студент (-ка) IV курсу, групи ТР-62
Мазуркевич Назар Віталійович

_____

Керівник:
кандидат технічних наук, доцент
Ходаковський Олексій Володимирович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

Мазуркевичу Назару Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи _Система аналітики для систем контролю та захисту_____
керівник роботи _Ходаковський Олексій Володимирович, к.т.н., доцент_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**
2. Строк подання студентом роботи ___7 червня 2020 року_____
3. Вихідні дані до роботи Тимчасове положення про організацію освітнього процесу в КПІ ім. Ігоря Сікорського_____
4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
Задача аналітики даних для системи контролю та доступу, існуючі системи аналітики безпеки та аналітики ефективності праці, засоби розробки програмного продукту, проектування структури даних для зберігання інформації, опис програмної реалізації, робота користувача з системою._____

5. Перелік ілюстративного матеріалу
Актуальність проблеми, завдання, результати аналізу існуючих аналогів, використані програмні засоби, структура системи, діаграма прецендетів,форма реєстрації та заповнення даних про вхід/вихід, форма додавання нового персоналу, таблиця помилок про вхід/вихід через систему, авторизації в системі, графік кількості відпрацьованих

годин до норми, графіки невідпрацьованих годин до норми та кількості виходів, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання "13" квітня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	13.04.2020	
2.	Вивчення та аналіз задачі	20.04.2020	
3.	Розробка архітектури та загальної структури системи	29.04.2020	
4.	Розробка структур окремих підсистем	06.05.2020	
5.	Програмна реалізація системи	17.05.2020	
6.	Оформлення пояснювальної записки	02.06.2020	
7.	Захист програмного продукту	10.06.2020	
8.	Передзахист	08.06.2020	
9.	Захист	16.06.2020	

Студент _____ Мазуркевич Н.В.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Ходаковський О.В.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою даної дипломної роботи є модуль розробки системи аналітики для систем контролю та захисту. Розроблений модуль автоматизації надає можливість додавати базові дані про прохід через систему, створювати та переглядати аналітичну інформацію про ефективність праці та безпеку системи. Розроблений модуль надає доступ до системи на основі рівнів доступу наданих кожному користувачу.

Дипломну роботу виконано на 80 аркушах, вона містить 56 ілюстрацій, 17 посилань на літературу та 3 додатки.

ABSTRACT

The purpose of the thesis is a module of data analysis development for electronic access control systems. The developed automation module provides an opportunity to enter basic data, create and view analytical information about work efficiency and security and safety of an access control system. The developed module grants access to the system based upon access levels provided to every user.

The thesis is completed on 80 sheets, it contains 56 illustrations, 17 references to literature and 3 annexes.

ЗМІСТ

ВСТУП

Існує багато переваг використання аналітики даних.В більшості випадків дані використовують для комерційний цілей, таких як:придабання та утримання клієнтів або маркетинг та покращення продукту.Однак,не так багато програмного забезпечення ,яке використовує аналітику даних, створюється для систем захисту або для покращення продуктивності та дисципліни коллективу.Програмне забезпечення для названих цілей існує, однак, унікальність даної роботи у розробці програмного забезпечення в першу чергу для навчальних установ та малих бізнесів беручі до уваги їхні первинні потреби.

Аналітика даних часто використовується в системах заисту та контролю для того щоб запобігти можливі випадки неконтрольованого доступу до інформаціїї або частини будівлі.Дані забезпечують інформацію про поведінкові та інші тенденції, які пов'язані із безпекою[1]. Наприклад, аналітична система може показувати графік роботи людини та на якому поверсі він / вона має працювати, а система контролю доступу може надавати дані про те, куди в будівлі вони йдуть. Аналіз даних може вказувати на аномалії, які можуть призвести до проблеми безпеки. Інтструменти аналітики даних дають відповіді на питання «Як часто працівник часто відвідує ділянку за межами звичного робочого середовища» , «Чи з’являється хтось на роботу в неробочий час» та багато інших.

Організація Fortune 500 має багато повідомлень про випадки дорогого обладнання, яке було викраденого з різних будівель навколо головного кампусу[2].Незважаючи на різні системи безпеки встановлені в будівлях організації, різні організаціїї не можуть забезпечити повний контроль та захист інформації й обладнання.Інструменти аналітики даних можуть допомогти у вирішення цієї проблеми, запобігаючи можливі спроби некотрольваного доступу.

Перша частина програмного продукту розроблена для відділу безпеки в компаніях малого бізнесу,державних та навчальних установах. Більшість з установ використовують пропускну систему встановленну на вході в будівлю, а також для

доступу до різних відділів будівлі.Так як працівники проходять через пропускну системи використовуючи картку, всі проходи додаються до бази даних дозволяючи відслідковувати анлітику системи захисту та анлітику ефективності працівників.Даний програмний продукт надає доступ до конкретних департаментів в залежності від позиції працівника та його/її повноважень, забороняє проходу через систему використовуючу одну й ту саму карту декілька разів запобігаючи спроби проведення не уповноважених осіб до різних відділів будівлі.Кількість проступок та інформацію про них зберігається,що дозволяє потім викорастати їх для оцінки ситеми безпеки в будівлі.

Друга частина оцінює результативність та показники ефективності роботи працівників. Людський капітал є основою будь-якої організації, а результативність роботи працівників має суттєвий вплив на успіхи самої організації. Фактично, дослідження показують, що п’ятивідсоткове збільшення ефективності роботи працівників пов'язане із зростанням річних доходів на три відсотки.Звичайні системи та методи огляду ефективності є застарілими, так як не дають чітких результатів, які дуже часто упередженні та суб’єктивні.Також такі методи не забезпечують своєчасного зворотного зв'язку для співробітників.Використання аналітики даних для оцінки ефективності працівників є сучасним та новітнім методом, який може покращити не тільки результати компанії але й відносини між самими працівниками.

Наприклад, у звіті консалтингової компанії PwC під назвою «Управління ефективністю в Індії - зміна бекконса» підкреслюється, що 52 відсотки організацій внесли або планують внести зміни в результативність праці працівників найближчим часом використовуючи аналітичний підхід[3].

Дуже часто компанії використовують інструменти аналітики даних для того щоб підібрати або оцінити персонал.Для цієї цілі програмний продукт часто розроблений та користувачами цього програмного продукту стають менеджери з підбору персоналу.Унікальність даної дипломної роботи полягаю в тому, що частина продукту яка відповідає за аналітику ефективності роботи працівників розроблена для бухгалтерів та аудиторів.Програма відслідковує таку інформіцію як:час прибутття на роботу,

кількість відпрацьованих годин в тижддень та порівнює іх до норми, загальний час проведений на обідніх перервах за місяц і так далі.Дана метрика не показую наскільки працівник є компетентим в своій області експертизи, але показую наскільки він/вона ефективно використовує час, дотримується норм порядку та його/її персональне відношення до окремих процесів робочого процесу.Далі ці дані використовується для оцінки ефективності того чи іншого працівника.Також премііїї або штрафи можуть бути накладеними на працівників в задежності від місячниї показників додаючи або віднімаючи фінансові активи від зарплати.Така система, з часом, може значно покращити результати організації та відносини в колективі.

Використання статистичного та аналітичного підходу є доволі поширеним інструментом для вирішення різного роду задач.Основною причиною популярності аналітики даних є той факт що даний інструмент дозволяє точно оцінити та відповісти на будь-які кількісні питання.В той же самий час, аналітика даних є незамінною частиною машинного навчання,яке,в свою чергу, застосовується майже в кожній сучасній області виробництва або дослідження[4].

1 ЗАДАЧА АНАЛІТИКИ ДАНИХ ДЛЯ СИСТЕМИ КОНТРОЛЮ ТА ДОСТУПУ

Осклільки була поставлена задача розробки аналітики даних для фізичної пропускної системи, дана дипломна робота та, зокрема, програмне забезпечення було розроблено.Даний проект розроблений для подальшого його інтегрування з фізичною пропускною системою розробленою на кафедрі «АПЕПС».

Даний програмний продукт поділений на дві частини:аналітика для систем контролю та захисту та аналітика ефективності роботи працівників.Метою цієї дипломної роботи є створення унікального програмного продукту, який можна використати як для цілей кафедри, так і для інших установ, які шукають аналітичний та статистичний підхід для вирішення конкретних проблем.

На вхід подається інформація про прохід через пропускну систему.Дані про час проходу, дату, користувача є ключовою інформацією, яка додається до бази даних.Дані додаються вручну,однак,в подальшому програмне забезпечення має підключатися до серверу та автоматично додавати інформацію про кожен прохід через систему до бази даних.

На вихід подається анлітика даних по помилках , які були зафіксовані під час проходу через систему та по спробам несанкціонованого доступу до певних частин будівлі.Також подається інформація про ефективність та дисципліну праці, а саме кількість відпрацьованих годин, виходи на перерви та запізнення.

Потенційими користувачами системи є працівники кафедри, а також працівники державних установ або малих бізнесів.Програмне забезпечення розроблене для установ з невеликою кільскістю працівників.

2 ІСНУЮЧІ СИСТЕМИ АНАЛІТИКИ БЕЗПЕКИ ТА АНАЛІТИКИ ЕФЕКТИВНОСТІ ПРАЦІ

Данна дипломна робота окреслює декілька задач.Базовою задачею є створення бази даних та програмного продукту для додання персоналу, працівників підприємства або державною установи та інформації про них.Так як даний проект розроблений з головним фокусом на використання його інтегравним з іншим продуктом розробленим на кафедрі «АПЕПС» університету НТУУ «КПІ», робота є доволі незалежною. Програмниий проект розроблений для конкретних потреб кафедри, але в той же час є універсальним для використання.Використання аналітики даних та статистичного підходу в даному програмному продукті для системи безпеки на кафедрі, робить дану роботу унікальною.

Студенти кафедри «АПЕПС» розробили фізичну пропускну систему,яку планують реалізувати та інсталювати на поверхах кафедри.Для доступу до конкретної аудиторії або сектору будівлі або навіть самої будівлі потрібно використовувати картку або бейдж.Це доволі популярний підхід для вирішення проблем безпеки.Пропускна система є унікальною сама по собі,так як розроблена в лабораторії, яка є частиною кафедри.Для повної реалізації системи та її тестування, було прийнято рішення розробити програмний продукт,який буде додавати інформацію про проходи через пропусну систему до бази даних та використовувати інформацію з бази для аналітики даних.Така аналітика даних та статистика дозволить забезпечити додатковий шар безпеки, захисту та контролю, показуючи всі випадки неправомірного доступу до певних частин будівлі або кімнат, а також всю підозрілу активність.Крім того було вирішено додати аналітику даних яка дозволить визначити еффективность праці та дисципліну працівників, беручи до уваги кількість відпрацьованих годин та виходів на перерви.Дана частина розроблена більше для колективів малих бізнесів, але також може успішно використовуватися для навчальних та інших установ.Для реалізації цілі кафедри, було розроблене дане

програмне забезпечення,яке, в свою чергу, є незалежною роботою від пропускної системи створеною раніше.

Дана задача включає в себе:

– аналіз предметної області;

– створення аналітики даних для систем контролю та захисту;

– створення аналітики даних для бухгалтерів та іншого персоналу;

– можливість виводити інформацію у зручному вигляді;

– можливість робити статистичні та аналітичні висновки на базі даних які були додані до бази даних.

2.1 Система «ACTsmart2 Software»

Існує велика кількість програм,які використовують аналітику даних для доступу та контролю, включаючи як додатковий захист для фізичних пропускних систем.Більшість з них є доволі схожими, адже в більшості випадків все зводиться до контролю інформацію про кожну фізичний взаємодію з систему безпеки[5].Однак,фінальні дані та показники які показанні в таких програмних продуктах можуть відрізнятись.Деякі системи показують всі зчитані помилки під час проходження через систему.Дуже час вони також інформують користувачів про підозрілу активність,таку як перебування в будівлі у вихідний день або зчитування однією картки декілька разів в дуже короткий проміжок часу.Така модель дуже схожа на підхід даного програмного продукту.Варто зазначити,що унікальність дипломного продукту в цьому випадку полягає в тому , що він розроблений для кафедри та фізичної прохідної системи, яка вже була розроблена.

Розглянемо приклад такої системи. На рисунку 2.1 можна побачити інтерфейс стандартної системи описаної вище. Система «ACTsmart2 Software» використовує рівні доступу, номер картки та ПІБ для кожної взаємодії з системою.Функціонально, така

система дуже схожа з програмним продуктом дипломної роботи, але в той же час є різними продуктами з різними цілями експлуатації. Система «ACTsmart2 Software» розроблена для великих компаній або організацій з великою кількістю персналу.Саме тому додаткові можливості інтерфейсу необхідні: фотографія обличчя людини та додаткова інформація про права доступу.Разом з додатковими можливостями ціна на такий програмний продукт збільшується і не є оптимальним вибором для кафедри.Так як система націлена на студентів та працівників кафедри, основний функціонал даного дипломного продукту є абсолютно достатнім для проблем безпеки кафедри або малого підприємства.Крім того, дипломний проект більше орієнтований на помилки та неправомірні проходження через систему,які успішно додаються до бази даних та показані в програмному продукті, для забезпечення безпеки, адже в більшості випадків користувачі системи є люди які працюють на кафедрі, тому не потребують строго конторолю та додаткового функіоналу.

Рисунок 2.1 – Інтерфейс системи ACTsmart2 Software

Можна зробити висновок ,що більшість програмних продуктів для систем контролю та захисту розроблені для комерційного використання.Ці програми мають добре розроблений інтерфейс та поставлені задачі, але, на жаль, більшість з них не адресують проблем та вимог малих підприємств або установ таких як кафедра або міська рада.

2.2 Система «Zoho»

Деякі програми надають перевагу використанню таблиць, графіків та статистичних методів.

Статистика - це форма математичного аналізу, яка використовує кількісно оцінені моделі, представлення та синопсису для заданого набору експериментальних даних або реальних досліджень. Статистика вивчає методології для збору, перегляду, аналізу та отримання висновків із даних.

Статистика використовується для узагальнення процесу, який аналітик використовує для характеристики набору даних. Якщо набір даних залежить від вибірки більшої сукупності, то аналітик може розробити інтерпретації щодо сукупності насамперед на основі статистичних результатів вибірки. Статистичний аналіз включає процес збору та оцінки даних, а потім узагальнення даних у математичній формі[6].

Окрім узагальнення даних, одним з основних завдань для якого використовують статистику є можливістю сформулювати логічні висновки та передбачити залежності між змінними на основі даних.

Статистика може бути потужним інструментом при виконанні завдань Data Science (DS). З точки зору високого рівня, статистика - це використання математики для виконання технічного аналізу даних. Основна візуалізація, така як діаграма, може дати вам інформацію високого рівня, але за допомогою статистики ми оперуємо даними

набагато більш орієнтованими на інформацію яка приведе нас до логічних висновків. Математика, яка є частиною статистики, допомагає нам сформувати конкретні висновки щодо наших даних, які слугуватимуть імперичним доказом.

Такі головні показники як середнє**,** мода та медіана є мірами центральної тенденції.

Середнє - це математичне середня величина для групи з двох чи більше цифр. Середнє значення для визначеного набору чисел може бути обчислено декількома способами, включаючи середнє арифметичне та геометричне середнє. Медіана є середнім числом в списку чисел. Для того щоб визначити медіану, потрібно упорядкувати всі точки даних і вибрати одне посередині або якщо є два середніх числа, беручи середнє значення цих двох чисел.Мода це найчастіше з чисел - тобто число, яке зустрічається найбільшу кількість разів.

Стандартне відхилення,яке часто представлене грецькою літерою сигма, є мірою поширення даних навколо середнього. Високе стандартне відхилення означає, що дані поширюються навкруги від середнього та покривають більше території, в той же час низьке сандартне відхилення є сигналом про те що більше даних узгоджуються із середнім та більш щільно розповсюджені. У портфоліо методів аналізу даних стандартне відхилення корисно для швидкого визначення дисперсії точок даних.

Так само, як і середнє значення, стандартне відхилення оманливе та не показує всіх результатів, якщо застосовувати тільки його. Наприклад, якщо в даних дуже важко знайти зразок або шаблон, то стандартне відхилення не дасть вам усієї необхідної інформації.

Однак графіки та статистичні показники набагато частіше використовують для оцінки роботи та показників ефективності працівників загалом.Розглянемо програм «Zoho», яка слугує гарним прикладом.

Програмний продукт “Zoho” дозволяє завантажувати різні дані про підприємство, працівників та будь-яку інформацію про фінансову звітність.Дані потім набувають табличного формату, які можна використовувати побудови графіків та чартів, знаходити

мода, медіна, середнє та багато іншого.Даний інструмент дозволяє аналізувати багато параметрів включаючи продуктивність,ефективність роботи та інше.Все залежить від інформації, яка буде додано до системи користувачем.Програма є доволі універсальною та популярною у використанні, адже функціонал обширний та дозволяє оперувати з аналітикою даних не тільки для оцінки працівників.Однак,таким чином продукт не відповідає конкретним потребам кафедри.Для малих установ така система скоріш за все не підійде також, тому що багато лишнього функціонала за який потрібно платити місячно.Такі програми, включаючи “Zoho”, потребують місячної підписки.Якщо користувач не збирається використовувати більшість можливостей програми та хоче задовольнити конкретні потреби, він/вона не бути використовувати таку програму.Розглянемо декілька основних осоливостей, які відрізняють даний програмний продукт від дипломного проекту.На рисуноках 2.2 та 2.3 можна побачити інтерфейс програми “Zoho”. Перш за все, дипломний продукт поділений на дві основні частини:аналітика системи контролю та доступу та аналітика продуктивності роботи базуючись на інформації проходів через пропусну систему доданою до бази даних.

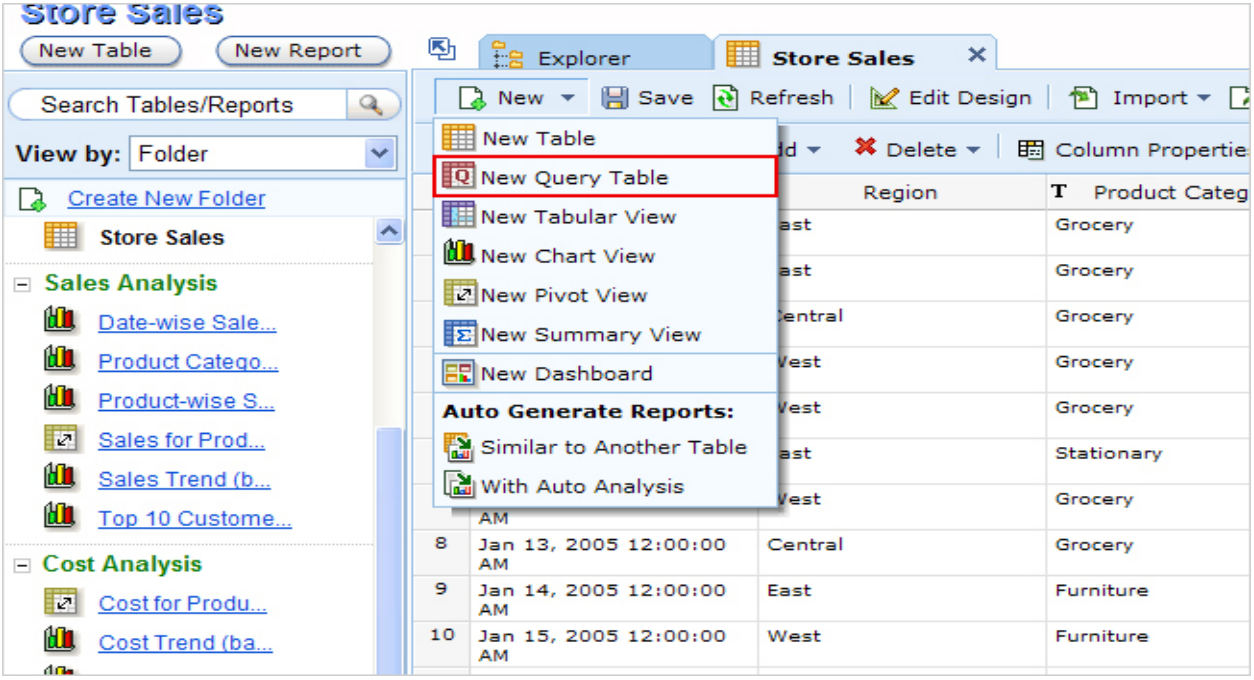


Рисунок 2.2 – Інтерфейс системи “Zoho”

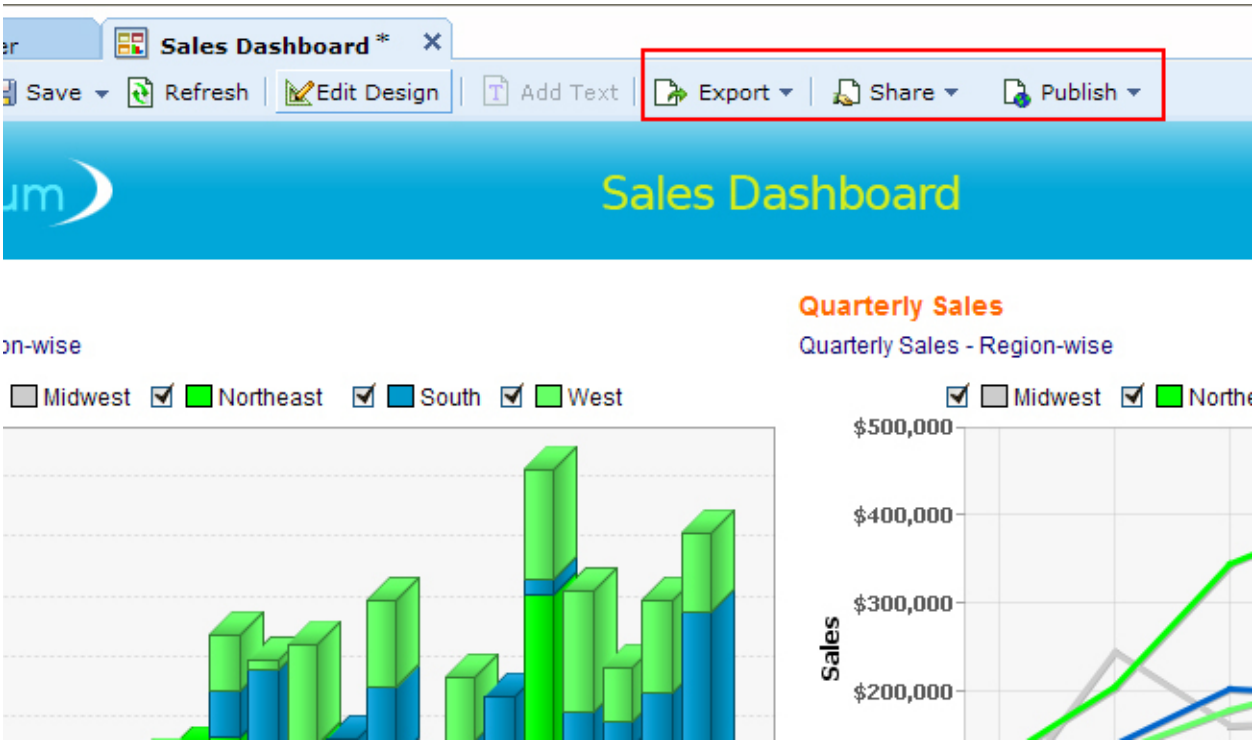


Рисунок 2.3 – Інтерфейс системи “Zoho”

Програма “Zoho”, незважаючи на те що є універсальною програмою для аналізування інформацію про результати роботи підприємства/установи, не була розробленою для роботи з системами безпеки та контролю.Інша значна різниця це використання конкретних даних отриманих від системи контролю для дипломного проекта: “Zoho” працює з даними які в основному пов’язані грошовими транзакціями, що знову ж таки не є потребою для кафедри або іншою установи, яка б хотіла використати дані системи контролю або безпеки для того щоб побачити наскільки ефективно працює перснал оперуючи даними про кількість відпрацьованих годин та кількість виходів.

2.3 Принципи системи контролю та ефективності

Можна зробити висновок, що більшість комерційних програм орієнтовані на корпорація або підприємства,які мають велику кількість персналу.Це твердження буде

правдивим як для програм розроблених для систем контролю так і ефективності роботи підприємства/установи.

Даний програмний продукт об’єднує дві різні системи,які в більшості випадків були б незалежними програмами для того щоб адресувати конкретні потреби кафедри «АПЕПС» а також різних установ.Проект розроблено незалежно від пропускної системи кафедри, але може бути доповненням як і до системи контролю кафедри або іншою установи.Даний продукт також не повинен бути залежним від будь якої системи контролю, адже інформація в базу даних про входи/виходи додається вручну.Однак, проект розроблений з цілю інтегрування до системи контролю, оскільки таким чином його використання буде найбільш ефективним.

В даному розділі було проаналізовано задачу дипломного проекта, а саме систем контролю та ефективності.Роботу було порівнянно з іншими комерційними програмами на ринку, основні різниці між ними були визначені.Ціль дипломного продукта була визначена, що дозволило віризнити її від інших схожих популярних продуктів на ринку та їх мети, зокрема відокремити користувачів системи.

3 ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

Для розробки програмного продукту було використано наступні засоби:

– середовище Visual Studio 2019;
– СУБД Ms SQL;
– Entity Framework;
– Система MVC

Перш за все для доступу до ресурсів необхідно створити базу даних.

За допомогою середовища Visual Studio 2019 мовою C# та з використанням бібліотеки Entity Framework було спроектовано та розроблено десктопний додаток, що дає змогу переглядати дані зручним для користувача способом.

3.1 Обгрунтування вибору технології Entity Framework

Entity Framework являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

Центральної концепцією Entity Framework є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами і їх наборами[7].

При цьому суті можуть бути пов'язані асоціативної зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою LINQ ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бд, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками[8].

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в БД. Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення (маппінга).

На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку.

Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних.

Рівень зіставлення (маппінга) служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць.

Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

3.2 Середовище розробки Visual Studio 2019

Середовище Visual Studio 2019 – це інтегроване середовище розробки програмного забезпечення від компанії Microsoft. За допомогою Visual Studio можна створювати додатки для Windows, iOS, Android та інших платформ. У Visual Studio включені інструменти не тільки для створення desktop додатків, але і web, мобільні і хмарні інструменти розробки. Є можливість написання коду на таких мовах як: C ++, C #, Visual Basic, F #, JavaScript, Python, TypeScript. Крім усього цього вона включає в себе

конструктори, редактори, відладчики, а також величезну кількість розширень для різних областей застосування - від PHP до ігор.

На рисунку 3.1 зображено інтерфейс Visual Studio 2019.

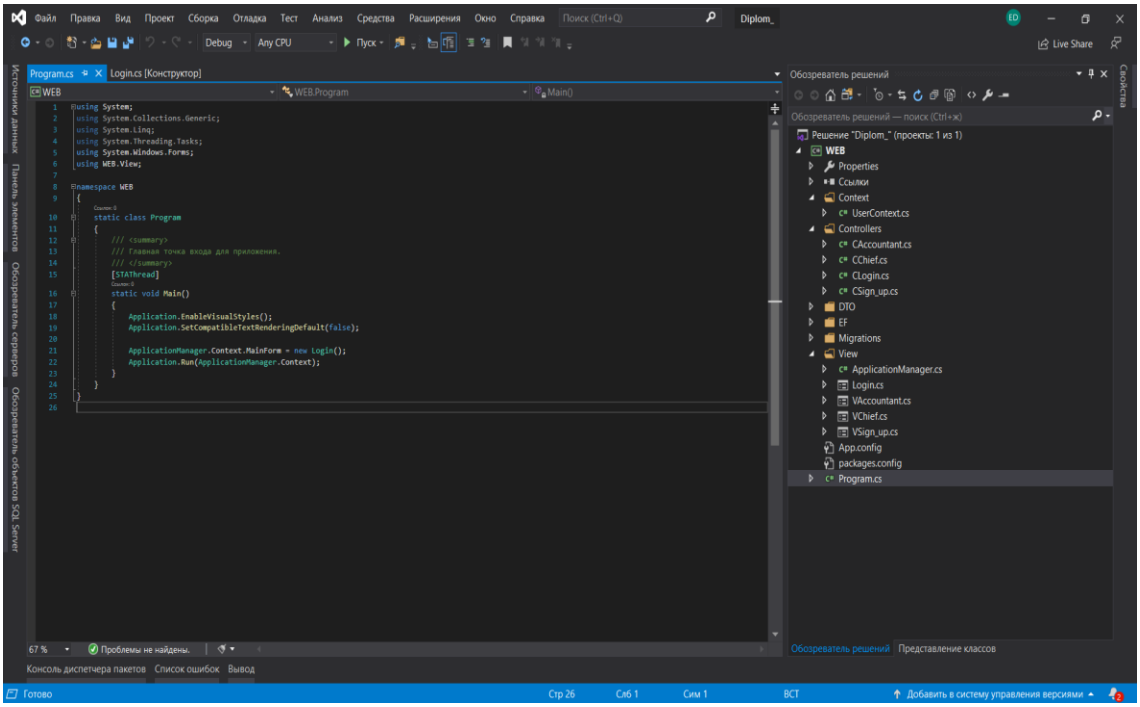


Рисунок 3.1 – інтерфейс Visual Studio 2019

Середовище Visual Studio 2019 року включає в себе наступні нові можливості і поновлення:

– випускається в трьох редакціях (раніше було чотири);
– крос-платформна підтримка мобільних пристроїв (Android, IOS, і Windows);
– покращення в C ++;
– зміни в налагодженні і діагностиці;
– платформа .NET Framework 4.8;
– покращена інтеграція Visual Studio і GitHub;
– і безліч інших.

3.3 Обгрунтування вибору платформи .NET Framework 4.8.

Платформа .NET Framework – це технологія, яка підтримує створення і виконання нового покоління додатків і веб-служб XML [9]. При розробці платформи .NET Framework були поставлені наступні цілі:

- оснащення платформи узгодженим об’єктно-орієнтованим середовищем програмування, що забезпечує збереження коду локально, дозволяє виконувати об’єктний код, який розміщено в мережі інтернет або для віддаленого доступу;
- створення середовища для виконання коду, яке оснащене системою контролю версій та зменшує проблеми при розгортанні програмного забезпечення;
- створення середовища для виконання коду, яке надає змогу безпечно виконувати код, включаючи код від невідомих постачальників;
- розробка середовища для виконання коду, яке поліпшує продуктивність виконання сценаріїв у середовищі та інтерпретування коду;
- встановлення єдиних парадигм розробки для різних видів клієнтських застосунків, наприклад, Windows і веб-додатки.

Саме за допомогою цієї платформи, написаний мною код конвертується в машинні операції, що дозволяє машині (ноутбуку) виконувати безперервну роботу, розробленої мною програми.

3.4 Обгрунтування вибору Microsoft SQL Server

Microsoft SQL Server — система управління базами даних, яка розробляється корпорацією Microsoft. Як сервер даних виконує головну функцію по збереженню та наданню даних у відповідь на запити інших застосунків, які можуть виконуватися як на

тому ж самому сервері, так і у мережі[10]. На рисунку 3.2 зображено інтерфейс програми Microsoft SQL Server.

Мова, що використовується для запитів — Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Багато років вдало конкурує з іншими системами керування базами даних.

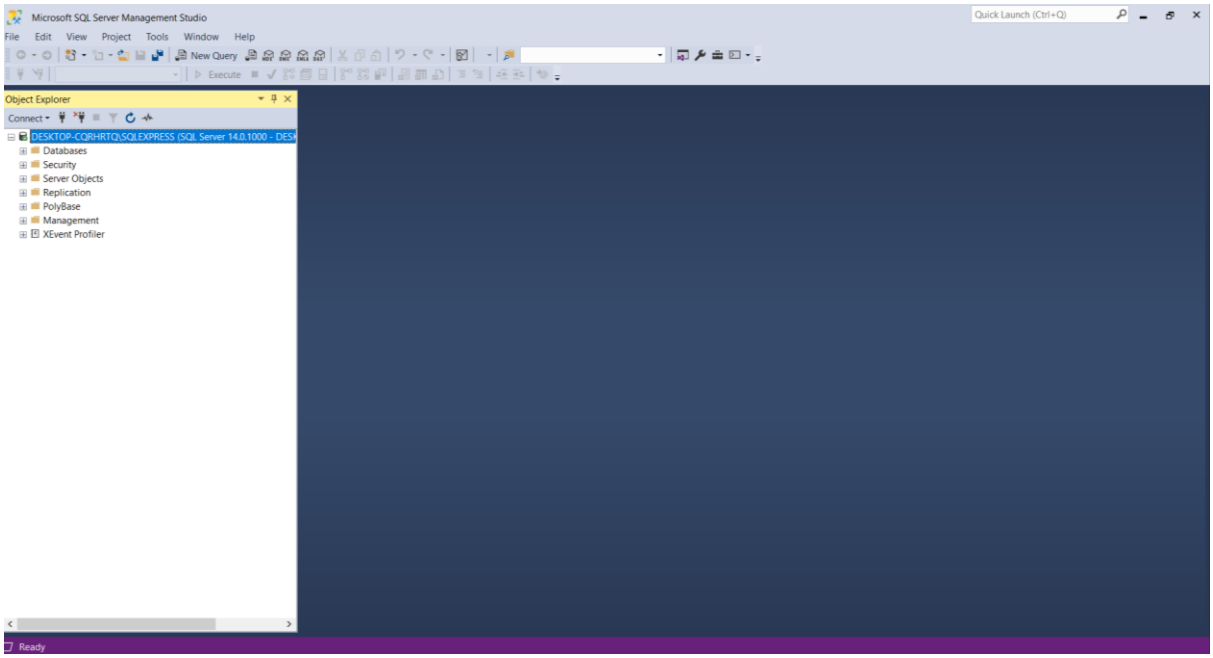


Рисунок 3.2 – Ітерфейс програми Ms sql Management server

SQL Server Management Studio (SSMS) - це програмне забезпечення, яке використовується для налаштування, управління та адміністрування всіх компонентів в Microsoft SQL Server. Інструмент включає в себе як редактори сценаріїв, так і графічні інструменти, які працюють з об'єктами та особливостями сервера.

3.5 Архітектурний шаблон Модель-Вигляд-Контролер

Моде́ль–вигляд–контро́лер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача[11].

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У рамках архітектурного шаблону модель–вигляд–контролер (MVC) програма поділяється на три окремі, але взаємопов'язані частини з розподілом функцій між компонентами. Модель (Model) відповідає за зберігання даних та їх структуру. Вигляд (View) відповідальний за представлення цих даних користувачеві, тобто інтерфейс програми. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача (зміна положення курсора миші, натискання кнопки, ввід даних в текстове поле) і передає дані у модель.

Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.

Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів

(представлень) однієї і тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.

Вигляд може бути:

- бути незалежними як від моделі, так і від контролера; або
- насправді бути контролером, і тому залежати від моделі.

Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

Модель інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних. Модель представляє дані, і не більше того. Модель не залежить від контролера або виду.

Вигляд може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції контролера входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Усунення непотрібних залежностей робить код простіший у обслуговуванні, оскільки він може бути використаний повторно без змін.Це надає можливість використовувати стабільний код, не вводячи в нього нових помилок.Основна перевага схеми дизайну MVC полягає в наступному: MVC робить класи моделей багаторазовими без змін[12].

Шаблон дизайну MVC вставляє клас контролера між виглядом та моделлю для усунення залежностей моделю та вигляд. При усуненні залежностей модель та, можливо, вигляд, можуть бути використані повторно без змін. Це робить реалізацію нових функцій та обслуговування легким. Користувачі швидко отримують стабільне програмне забезпечення, компанія економить гроші, а розробники мають можливість працювати з більш зручною технологією.

4 ПРОЕКТУВАННЯ СТРУКТУРИ ДАНИХ ДЛЯ ЗБЕРІГАННЯ ІНФОРМАЦІЇ

Для зручної взаємодії з даними буде використовуватись MsSql. База даних буде знаходитись на певному сервері, і можна буде в проекті легко використовувати всі необхідні дані. За допомогою Entity база даних буде гнучкою, її легко можна розширювати або змінювати.

4.1 Проектування архітектури головних таблиць

Головною таблицею виступає Staffs (персонал). Таблиця містить всю необхідну інформацію про персонал організації. На рисунку 4.1 зображена таблиця “Staffs”.

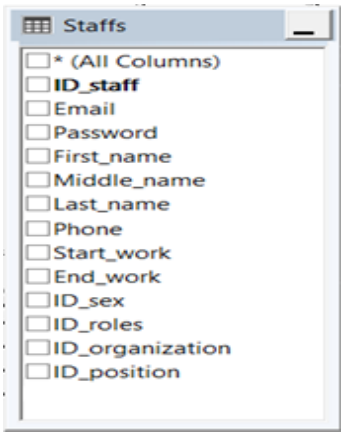


Рисунок 4.1 – Таблиця “Staffs”

Інформація про окремі стовбці таблиці:

– ID_staff – оригінальний номер запису в таблиці, який однозначно ідентифікує певного користувача. Всі таблиці в БД мають такий стовпець, несучи однакове ідейне навантаження. У всіх випадках полю призначене властивість первинного ключа;

- Email – текстове поле, що містить інформацію про емейл для авторизації в системі;
- Password – текстове поле, що містить інформацію про пароль для авторизації в системі;
- First_name – текстове поле, що містить ім’я персоналу
- Middle_name – текстове поле, що містить прізвище;
- Last_name – текстове поле, що містить інформацію про ім’я побаткові;
- Phone – поле, що містить інформацію про номер телефону;
- Start_work – поле, що містить інформацію про час початку роботи персоналу.
- End_work – поле, що містить інформацію про час закінчення робочого дня;
- ID_sex – вторинний ключ, що вказує на статичну таблицю стать (чоловіча, жіноча). Ці значення містяться у відповідній таблиці-словнику;
- ID_roles – вторинний ключ, що вказує на статичну таблицю ролей (керівник, користувач, бухгалтер, охорона). Ці значення містяться у відповідній таблиці-словнику;
- ID_organization – вторинний ключ, що вказує на організацію, до якої відноситься персонал. Ці значення містяться у відповідній таблиці-словнику;
- ID_position – вторинний ключ, що вказує на посаду на якій працює персонал. Ці значення містяться у відповідній таблиці-словнику;

Інформацію на які посилається “Staff” за допомогою зовнішніх ключів, описано на рисунках 4.2-4.5.

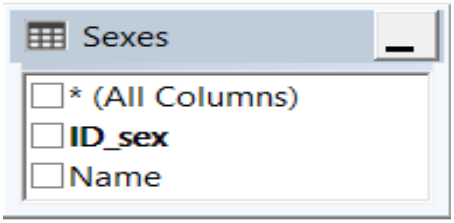


Рисунок 4.2 – Таблиця Sexes

Таблиця є статичною і містить інформацію про стать (чоловіча, жіноча). За це відповідає текстове поле Name.

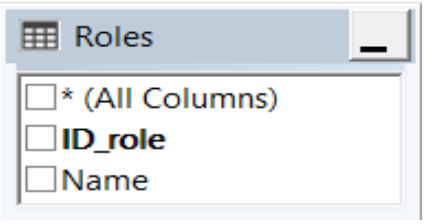


Рисунок 4.3 – Талиця “Roles”

Таблиця є статичною і містить інформацію про роль персоналу (керівник, бухгалтер, користувач, охорона). За це відповідає текстове поле Name.

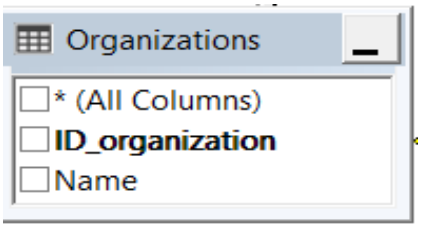


Рисунок 4.4 – Таблиця “Organizations”

Таблиця містить інформацію про назву організації у текстовому полі Name.

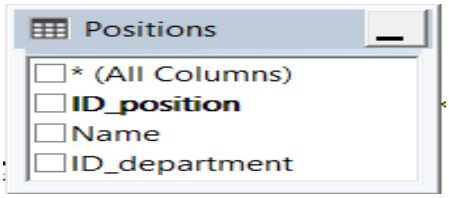


Рисунок 4.5 – Таблиця “Positions”

Інформація про окремі стовбці таблиці:

– Name – текстове поле, що містить інформацію про назву посади на якій працює персонал;
– ID_department – вторинний ключ, що вказує на таблицю відділу до якого відноситься посада. Ці значення містяться у відповідній таблиці-словнику.

На рисунку 4.6 зображена таблиця на яку посилається “Positions” за допомогою зовнішнього ключа;

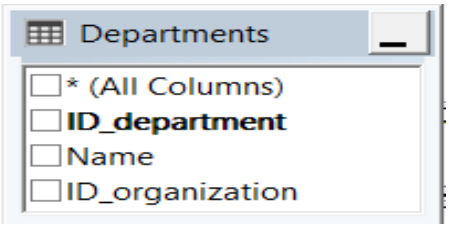


Рисунок 4.6 – таблиця “Departments”

Інформація про окремі стовбці таблиці:

– Name – текстове поле, що містить інформацію про назву відділу;
– ID_organization – вторинний ключ, що вказує на таблицю організації до якої відносяться відділи. Ці значення містяться у відповідній таблиці-словнику.

4.2 Проектування архітектури зв’язків

Ця частина є найголовнішою в системі, оскільки була задача створення аналітики про вхід/вихід персоналу з приміщення. На рисунку 4.7 зображено таблицю “Rooms”.

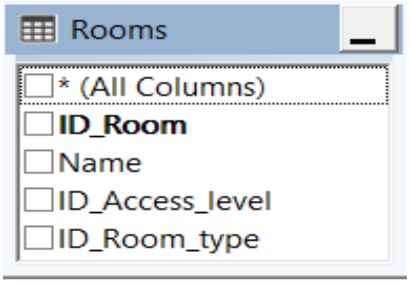


Рисунок 4.7 – Таблиця “Rooms”

Інформація про окремі стовбці таблиці:

– Name – текстове поле, що містить інформацію про номер приміщення;

– ID_Access_level – вторинний ключ, що вказує на таблицю рівня доступу до якого відноситься приміщення. Ці значення містяться у відповідній таблиці-словнику.
– ID_Room_type – вторинний ключ, що вказує на таблицю типу приміщення (зал для конференцій, кабінет, спортзал). Ці значення містяться у відповідній таблиці-словнику.

Інформацію на які посилається “Rooms” за допомогою зовнішніх ключів, описано на рисунках 4.8-4.9.

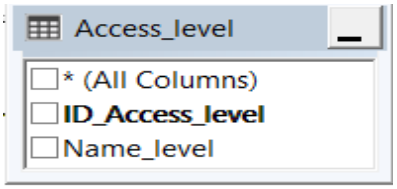


Рисунок 4.8 – Таблиця “Access_level”

Таблиця містить інформацію про назву рівня доступу у текстовому полі Name_level.

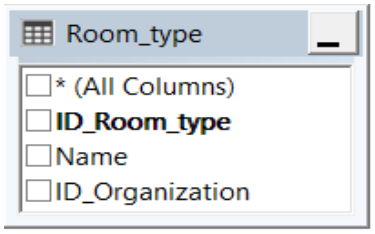


Рисунок 4.9 – Таблиця “Room_type”

Інформація про окремі стовбці таблиці:

– Name – текстове поле, що містить інформацію про тип приміщення;
– ID_Organization – вторинний ключ, що вказує на таблицю організації до якої відноситься тип приміщення. Ці значення містяться у відповідній таблиці-словнику.

Для забезпечення зв’язку між персоналом і можливістю входу в приміщення використовується таблиця “Additional_access_staffs”. Оскільки кожний працівник може

одночасно мати декілька рівнів доступу до різних приміщень, то ця таблиця виступає проміжною для забезпечення зв’язку багато-до-багатьох[13]. На рисунку 4.10 зображено таблицю “Additional_access_staffs”.

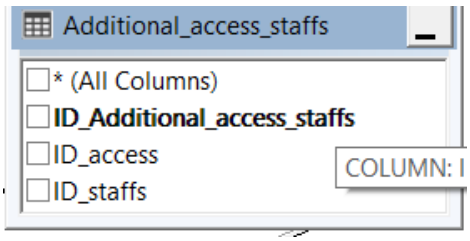


Рисунок 4.10 – Таблиця “Additional_access_staffs”

Інформація про окремі стовбці таблиці:

– ID_access – вторинний ключ, що вказує на таблицю рівнів доступу. Ці значення містяться у відповідній таблиці-словнику описаній вище.
– ID_staffs – вторинний ключ, що вказує на таблицю персоналу. Ці значення містяться у відповідній таблиці-словнику описаній вище.

Таблиця “Entrances” зберігає інформацію про вхід/вихід певного користувача в приміщення зображено на рисунку 4.11.

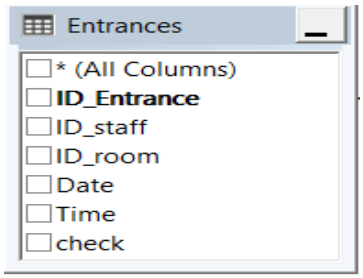


Рисунок 4.11 – Таблиця “Entrances”

Інформація про окремі стовбці таблиці:

– ID_staff – вторинний ключ, що вказує на таблицю персоналу. Ці значення містяться у відповідній таблиці-словнику описаній вище.
– ID_room – вторинний ключ, що вказує на таблицю приміщень. Ці значення містяться у відповідній таблиці-словнику описаній вище.

- Date – поле, що містить інформацію про дату входу/виходу в певне приміщення.
- Time – поле, що містить інформацію про час входу/виходу в приміщення.
- Check – поле, що містить інформацію проте, що користувач війшов, чи вийшов з приміщення.

Таблиця “Violations” містить інформацію про помилки входу в приміщення (користувач пробує зайти в приміщення до якого немає доступу, користувач пробує зайти в приміщення не вийшовши з попереднього або пройшов через систему в неробочий час). На рисунку 4.12 зображено таблицю “Violations”.

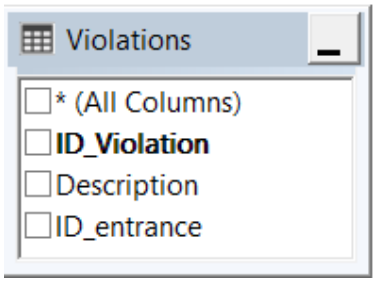


Рисунок 4.12 – таблиця “Violations”

Інформація про окремі стовбці таблиці:

- Description – текстове поле, що містить опис помилки.
- ID_entrance – вторинний ключ, що вказує на таблицю входу/виходу з приміщення. Ці значення містяться у відповідній таблиці-словнику описаній вище.

На рисунку 4.13 зображено всю концептуальну модель бази даних.

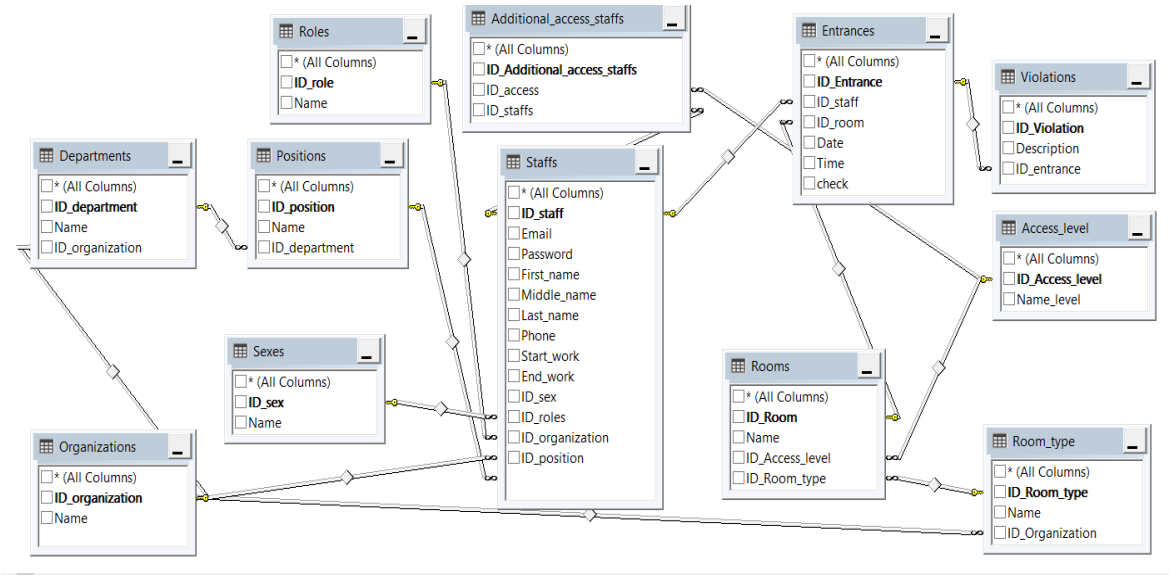


Рисунок 4.13 – Концептуальна модель бази даних

Можна побачити, всі зв’язки і їхні відношення між таблицями, загальну структуру бази даних.

Отже, у даному розділі було спроектовано та частково реалізовано архітектуру системи. Спроектовані структури даних для зберігання інформації у всіх структурних елементах застосунку. Спроектовані окремі моделі даних для бази даних MsSql, для використання даних із сховища. Модуль був розроблений таким чином, щоб можливо було легко доповнювати і розширювати базу даних.

5 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення системи було використано середовище: Visual Studio 2019. Перша частина дипломної роботи – створення бази даних за допомогою Entity,

підключення до Microsoft SQL Server. Друга частина – розробка інтерфейсу для різних ролей системи. Клієнтський додаток написаний мовою C#.

5.1 Створення моделі даних

Першим кроком для роботи з Entity є – інсталювання його до проекту. Використаємо інтерфейс середовища Visual Studio. Перейдемо до пункту«Управління пакетами NuGet», в пошуку знайдемо EntityFramework і інсталюємо до проекту. На рисунку 5.1 зображено інтерфейс для підключення Entity.

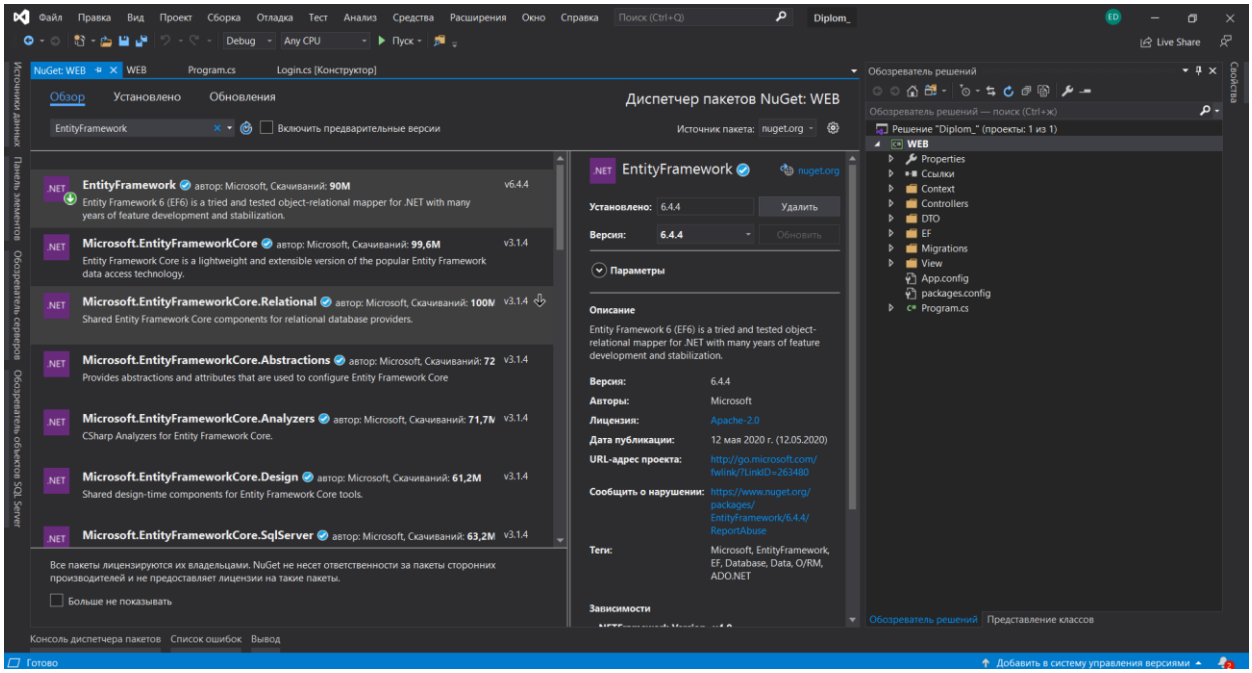


Рисунок 5.1 – інтерфейс підключення Entity

Модель являє звичайний клас на мові C #. Всі моделі тут мають набір властивостей, що описують реальні властивості об'єкта. У той же час при створенні

моделей слід дотримуватися деяких умовності. Оскільки ми будемо використовувати для зберігання моделей базу даних SQL Server, то для маніпуляції над об'єктами в базі даних нам треба визначити для них первинний ключ (Primary Key), який виконує роль універсального ідентифікатора об'єкта. Тому першим властивістю в кожної моделі йде властивість Id, призначене для зберігання первинного ключа. Визначення ключа за допомогою атрибута Key, встановленим над потрібною властивістю. Зв'язок один-до-багатьох реалізується, якщо одна модель зберігає посилання на один об'єкт іншої моделі. На рисунку 5.2 зображено реалізацію зв’язку один до багатьох.

```
[ForeignKey("Sex")]
Ссылок: 6
public int ID_sex { get; set; }
[ForeignKey("Role")]
Ссылок: 4
public int ID_roles { get; set; }
[ForeignKey("Organization")]
Ссылок: 21
public int ID_organization { get; set; }
[ForeignKey("Position")]
Ссылок: 6
public int? ID_position { get; set; }


Ссылок: 2
public Role Role { get; set; }
ссылка: 1
public Organization Organization { get; set; }
Ссылок: 5
public Sex Sex { get; set; }
Ссылок: 13
public Position Position { get; set; }
```

Рисунок 5.2 – Реалізація зв’язку один до багатьох

Поле ID_sex є ключем типу ForeignKey. Це поле зберігає індекс із таблиці sex, для забезпечення зв’язку.

Зв’язок багато до багатьох реалізовується за допомогою додаткового класу. На рисунку 5.3 зображено додатковий клас, який зв’язує багато до багатьох класи доступу і персоналу.

```
public class Additional_access_staffs
{
    [Key]
    public int ID_Additional_access_staffs { get; set; }
    public Staff Staffs { get; set; }
    public Access_level Access { get; set; }


    [ForeignKey("Access")]
    public int ID_access { get; set; }
    [ForeignKey("Staffs")]
    public int ID_staffs { get; set; }
}
```

Рисунок 5.3 – Приклад додаткового класу, який зв’язує багато до багатьох класи доступу і персоналу

На рисунку 5.4 зображено всі класи-об’єкти бази даних.

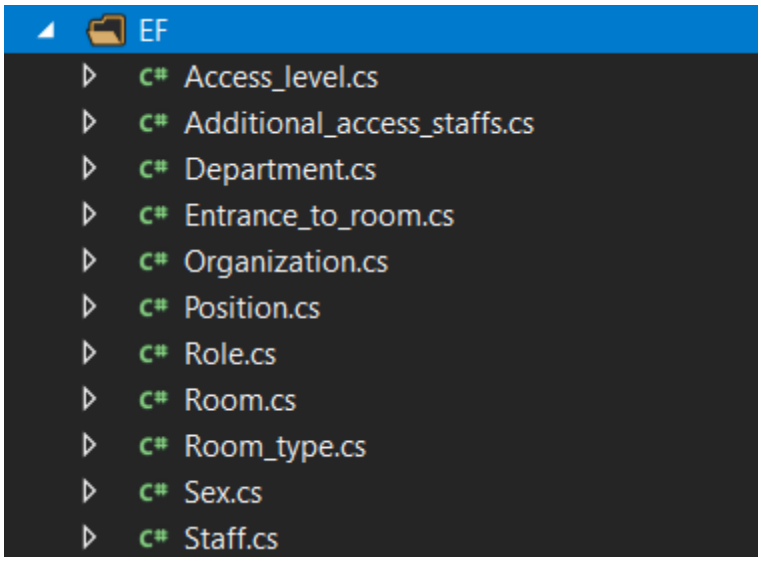
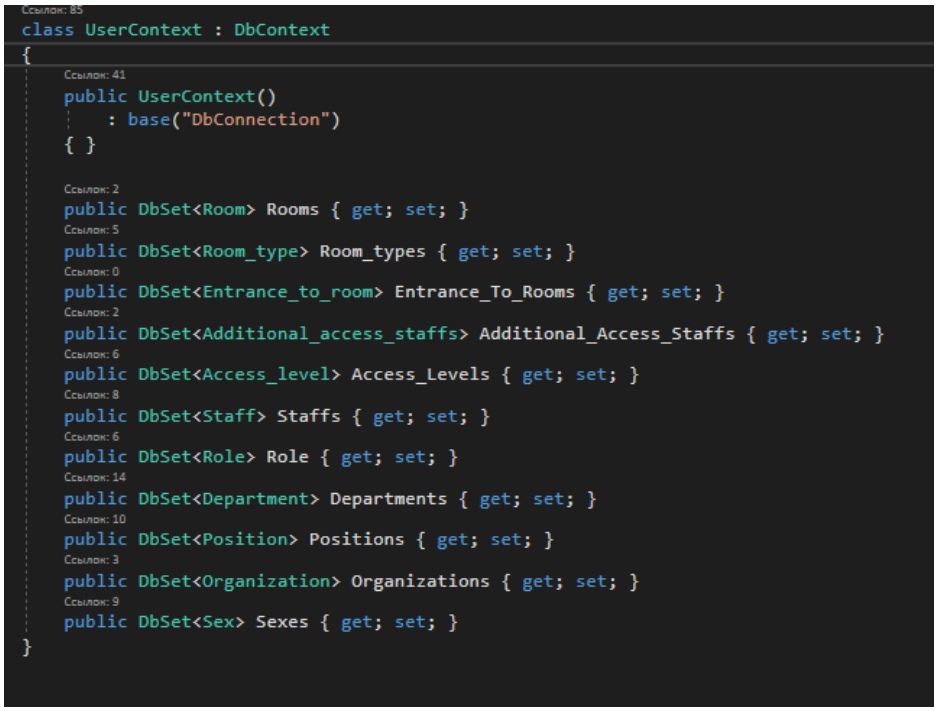


Рисунок 5.4 – Папка зі всіма класами-таблицями у проекті

Після цього створимо контекст даних. Контекст даних використовує EntityFramework для доступу до БД на основі деякої моделі. Отже, додамо в папку Context новий клас UserContext. На рисунку 5.5 зображено клас UserContext.



```
Ссылок: 85
class UserContext : DbContext
{
    Ссылок: 41
    public UserContext()
        : base("DbConnection")
    { }

    Ссылок: 2
    public DbSet<Room> Rooms { get; set; }
    Ссылок: 5
    public DbSet<Room_type> Room_types { get; set; }
    Ссылок: 0
    public DbSet<Entrance_to_room> Entrance_To_Rooms { get; set; }
    Ссылок: 2
    public DbSet<Additional_access_staffs> Additional_Access_Staffs { get; set; }
    Ссылок: 6
    public DbSet<Access_level> Access_Levels { get; set; }
    Ссылок: 8
    public DbSet<Staff> Staffs { get; set; }
    Ссылок: 6
    public DbSet<Role> Role { get; set; }
    Ссылок: 14
    public DbSet<Department> Departments { get; set; }
    Ссылок: 10
    public DbSet<Position> Positions { get; set; }
    Ссылок: 3
    public DbSet<Organization> Organizations { get; set; }
    Ссылок: 9
    public DbSet<Sex> Sexes { get; set; }
}
```

Рисунок 5.5- клас UserContext

Щоб створити контекст, нам треба успадкувати новий клас від класу DbContext. Властивості зразок public DbSet <Staff> Staffs{get; set; } Допомагають отримувати з БД набір даних певного типу (наприклад, набір об'єктів Staff).

Хоча ми будемо використовувати базу даних, але створювати явним чином ми її не будемо. За нас все зробить EntityFramework. Це так званий підхід Code First - у нас є моделі, і по ним фреймворк буде створювати таблиці в базі даних.

Розробка Entity Framework Code First допомагає нам спершу визначити модель домену, а потім делегувати процес створення бази даних Entity Framework на основі доменної моделі. Це називається Design Driver Design, де розробники визначатимуть модель домену як POCO-класи, а EF створить таблиці баз даних на основі класів POCO[14].

Існують, головним чином, два способи здійснення кодуючих міграцій,

-Міграції на основі коду

-Автоматична міграція

Для міграцій Entity Framework пакує набір команд, які виконуються з консолі диспетчера пакетів.

І на завершення роботи над модельної частиною встановимо рядок підключення. Для цього відкриємо файл App.config, знайдемо секцію configSections і відразу після неї вставимо секцію connectionStrings. На рисунку 5.6 зображено строку підключення до БД.

```
<connectionStrings>
  <add name="DBConnection" connectionString="data source=DESKTOP-CQRHRTQ\SQLEXPRESS;Initial Catalog=SS__7;Integrated Security=True;"
providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Рисунок 5.6 – підключення до БД

Data source ім’я серверу на якому буде знаходитися база даних, initial Catalog вказує на назву бази даних.

Для створення бази даних на сервері потрібно перейти у вікно: «Консоль диспечера пакетів».

При розробці нової програми ваша модель даних часто змінюється, і при кожному таку зміну вона порушує синхронізацію з базою даних. Потім при кожній зміні моделі даних - додавання, видалення або зміну класів сутностей або зміні класу DbContext ви можете видалити базу даних, після чого Entity Framework створює нову, яка відповідає даній моделі, і заповнює її тестовими даними.

Цей спосіб для забезпечення синхронізації бази даних з моделлю даних добре працює до розгортання програми в робочому середовищі. Коли додаток виконується в робочому середовищі, воно зазвичай зберігає дані, які ви хочете зберегти, і вам небажано втрачати все при кожній зміні, наприклад при додаванні нового стовпчика. Функція міграцій Entity Framework вирішує цю проблему, дозволяючи Entity Framework оновити схему бази даних замість створення бази даних за допомогою команди Add-Migration NameMigration. Після її виконання створиться папка Migrations у проекті.Потрібно ще оновити саму базу даних за допомогою команди Update-Database. Можна перевірити саму базу даних у середовищі SQL Server: Management Studio, зображено на рисунку 5.7.

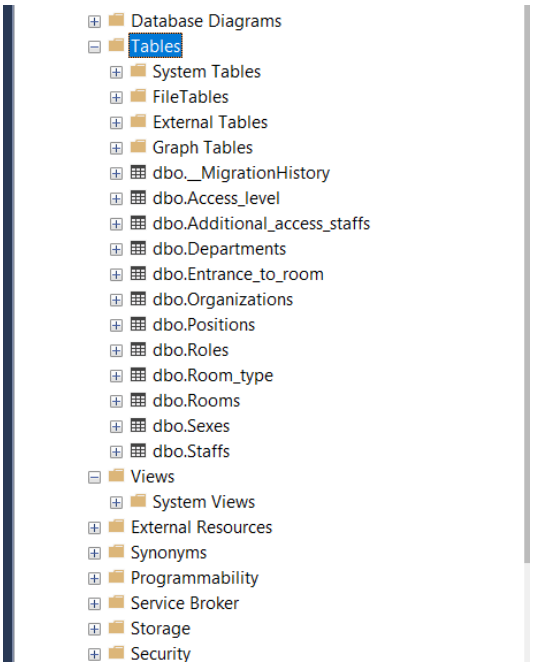


Рисунок 5.7 – БД в SQL Server: Management Studio середовищі

Перший рівень проекту створений, перейдемо до другого рівня: Controller. Який буде зєднувати між собою представлення і модель даних.

5.2 Створення контролерів і DTO

Рівень представлення не може безпосередньо отримувати дані з бази даних. В даному випадку Controller виступатиме в ролі посередника між двома рівнями. Отже, додамо в проект папку, яку назвемо DTO[15]. Визначимо в ній новий клас StaffDTO.

Через цей клас ми будемо передавати об'єкти Staff між рівнями. Клас StaffDTO повинен містити тільки ті дані, які ми збираємося передати на рівень представлення або, навпаки, отримати з цього рівня. Тобто це те, що називається Data Transfer Object - спеціальна модель для передачі даних[16]. Приклад реалізації зображено на рисунку 5.8.

```
public List<StaffDTO> GetAllStaff(int id)
{
    using (UserContext db = new UserContext())
    {
        List<Staff> rtn = db.Staffs.Where(x => x.ID_organization == id).ToList();

        for (int i = 0; i < rtn.Count; ++i)
        {
            rtn[i].Sex = db.Sexes.Find(rtn[i].ID_sex);
            rtn[i].Position = db.Positions.Find(rtn[i].ID_position);
            rtn[i].Position.Department = db.Departments.Find(rtn[i].Position.ID_department);
        }

        List<StaffDTO> list_ = new List<StaffDTO>();
        for (int i = 0; i < rtn.Count; ++i)
        {
            list_.Add(new StaffDTO
            {
                ID_staff = rtn[i].ID_staff,
                First_name = rtn[i].First_name,
                Middle_name = rtn[i].Middle_name,
                Last_name = rtn[i].Last_name,
                Sex = rtn[i].Sex.Name,
                Department = rtn[i].Position.Department.Name,
                Position = rtn[i].Position.Name,
                Phone = rtn[i].Phone,
                Start_work = rtn[i].Start_work,
                End_work = rtn[i].End_work,
                Email = rtn[i].Email,
                Password = rtn[i].Password,
            });
        }

        return list_;
    }
}
```

Рисунок 5.8 – Об’єднання даних для представлення

Велику роль в додатку грає валідація даних. Здебільшого за валідацію відповідає саме Controller. У контролері ми легко можемо провалідувати модель через об'єкт DTO, реалізацію зображено на рисунку 5.9.

```
string Error = string.Empty;
UserDTO user = new UserDTO { Email = email, Password = password };

var results = new List<ValidationResult>();
var context = new ValidationContext(user);

if (!Validator.TryValidateObject(user, context, results, true))
{
    foreach (var error in results)
    {
        Error += error.ErrorMessage + "\n";
    }

    return Error;
}
```

Рисунок 5.9 – Код контроллера який відповідає за валідацію даних у представленні при авторизації

Для вказання атрибутів валідації, можна в DTO використовувати [Required], [EmailAddress] і тд, зображено на рисунку 5.10.

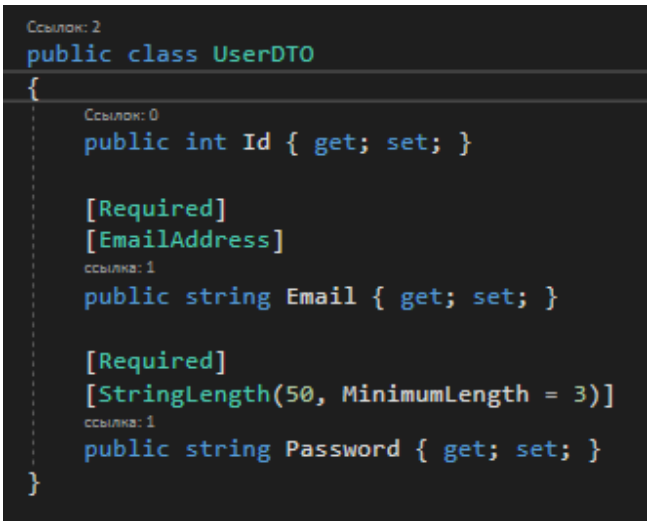


Рисунок 5.10 – Приклад класу UserDTO

Але хоча даний клас багато в чому схожий з визначення на клас Staff, це необов'язкова умова. Клас StaffDTO повинен містити тільки ті дані, які ми збираємося передати на певний рівень.

В залежності від атрибуту, буде накладуватись певний стиль запису, якщо Email, то поле повинно містити текст перед і після символу @. Якщо stringLength(50, MinimumLength = 3) – це значить, що пароль повинен бути неменше 3 і небільше 50.

Для забезпечення безпеки реєстрації нових користувачів використовується метод генерації пароль. На рисунку 5.11 зображено реалізацію генерації паролю.

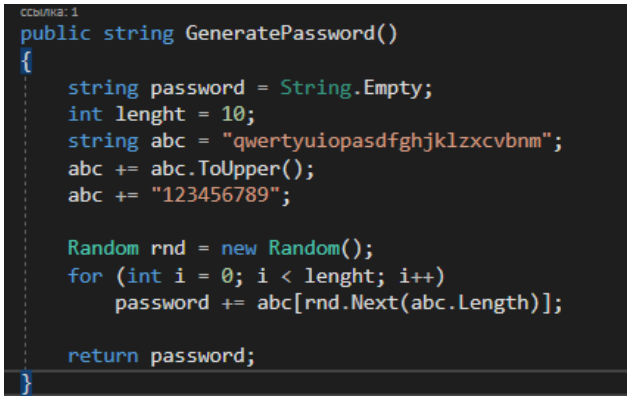
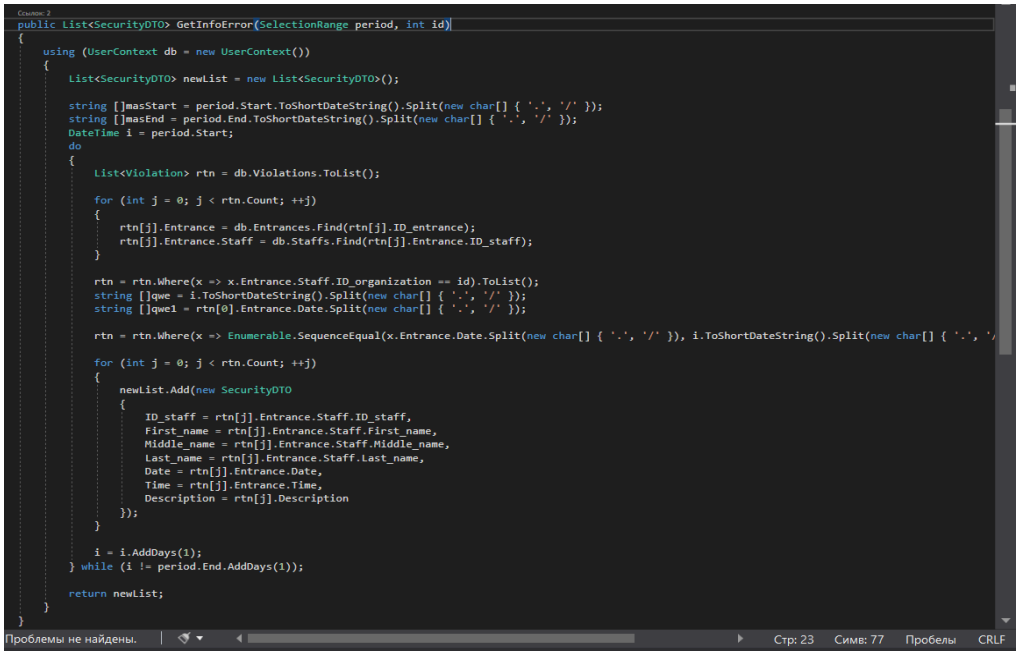


Рисунок 5.11 – Реалізація генерації паролю

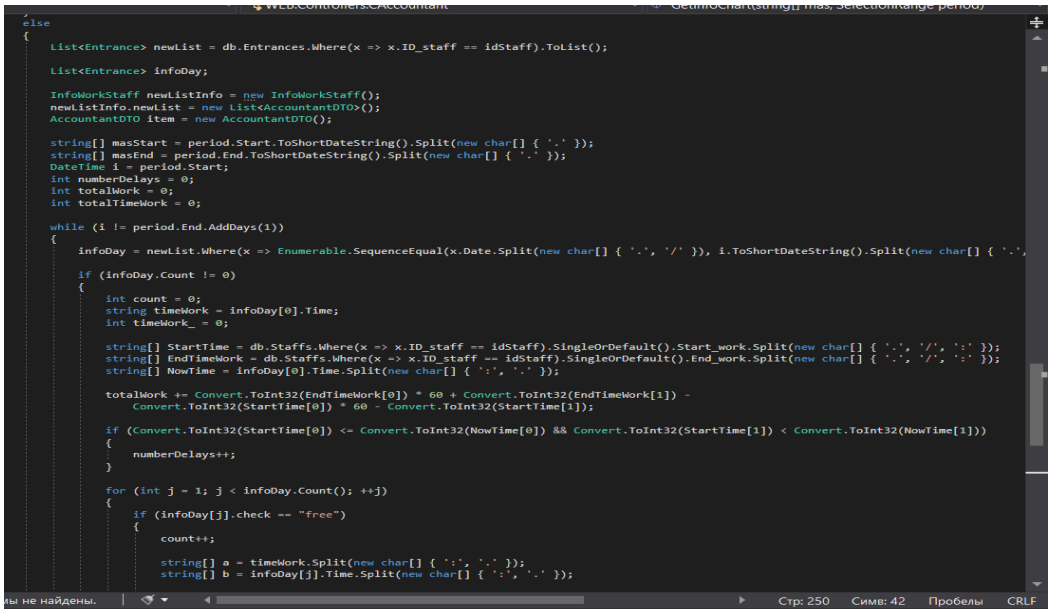
Використано загальну строку даних, яка буде використовуватись для генерації паролю. Сам механізм генерації заключається в тому, що генерується випадкове число яке відповідає індексу строки. Дана процедура виконується кількість разів прописану у length. Прикладом є: 9YpgQqNjMl, TaMEWGHQJo, HoTno5y1nv. На рисунку 5.12 зображено реалізацію форми охорони.



```
Ссылок: 2
public List<SecurityDTO> GetInfoError(SelectionRange period, int id)
{
    using (UserContext db = new UserContext())
    {
        List<SecurityDTO> newList = new List<SecurityDTO>();

        string []masStart = period.Start.ToShortDateString().Split(new char[] { '.', '/' });
        string []masEnd = period.End.ToShortDateString().Split(new char[] { '.', '/' });
        DateTime i = period.Start;
        do
        {
            List<Violation> rtn = db.Violations.ToList();

            for (int j = 0; j < rtn.Count; ++j)
            {
                rtn[j].Entrance = db.Entrances.Find(rtn[j].ID_entrance);
                rtn[j].Entrance.Staff = db.Staffs.Find(rtn[j].Entrance.ID_staff);
            }

            rtn = rtn.Where(x => x.Entrance.Staff.ID_organization == id).ToList();
            string []qwe = i.ToShortDateString().Split(new char[] { '.', '/' });
            string []qwe1 = rtn[0].Entrance.Date.Split(new char[] { '.', '/' });

            rtn = rtn.Where(x => Enumerable.SequenceEqual(x.Entrance.Date.Split(new char[] { '.', '/' }), i.ToShortDateString().Split(new char[] { '.', '/

            for (int j = 0; j < rtn.Count; ++j)
            {
                newList.Add(new SecurityDTO
                {
                    ID_staff = rtn[j].Entrance.Staff.ID_staff,
                    First_name = rtn[j].Entrance.Staff.First_name,
                    Middle_name = rtn[j].Entrance.Staff.Middle_name,
                    Last_name = rtn[j].Entrance.Staff.Last_name,
                    Date = rtn[j].Entrance.Date,
                    Time = rtn[j].Entrance.Time,
                    Description = rtn[j].Description
                });
            }

            i = i.AddDays(1);
        } while (i != period.End.AddDays(1));

        return newList;
    }
}
```


Рисунок 5.12 – Реалізація виводу даних для охорони

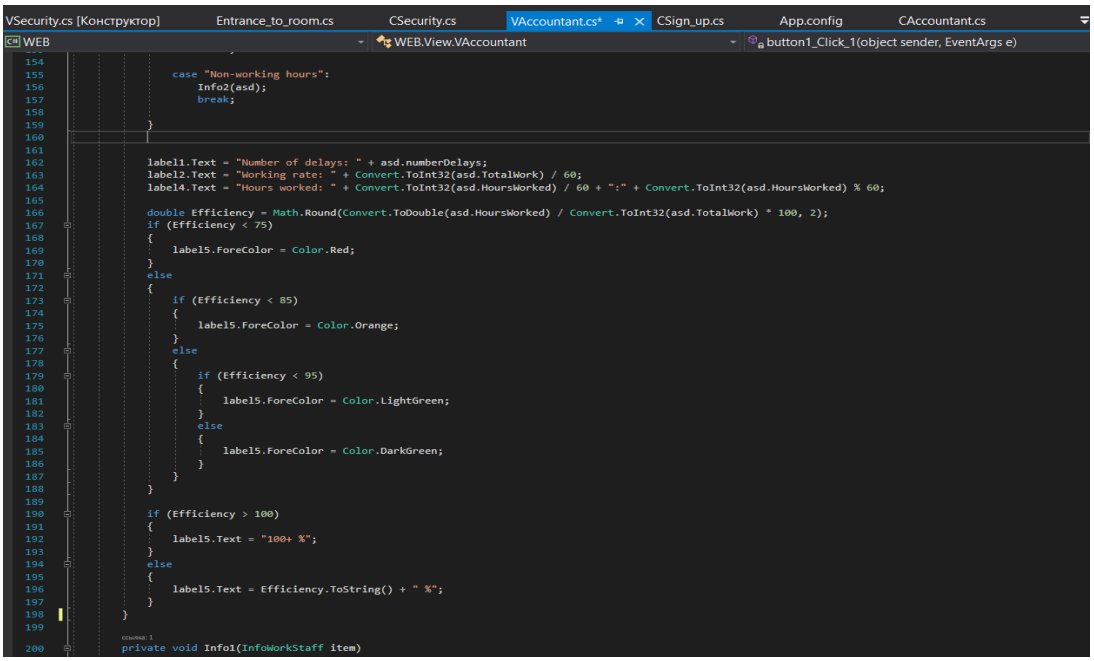
Дані отримуються безпосередньо з бази даних. Для зручного використання можна обирати певний період для відображення даних по ньому. Це робить систему більш гнучкою і комфортною для користування. На рисунку 5.13 зображено реалізацію виводу інформації для бухгалтера.



```
else
{
    List<Entrance> newList = db.Entrances.Where(x => x.ID_staff == idStaff).ToList();

    List<Entrance> infoDay;

    InfoWorkStaff newListInfo = new InfoWorkStaff();
    newListInfo.newList = new List<AccountantDTO>();
    AccountantDTO item = new AccountantDTO();

    string[] masStart = period.Start.ToShortDateString().Split(new char[] { '.' });
    string[] masEnd = period.End.ToShortDateString().Split(new char[] { '.' });
    DateTime i = period.Start;
    int numberDelays = 0;
    int totalWork = 0;
    int totalTimeWork = 0;

    while (i != period.End.AddDays(1))
    {
        infoDay = newList.Where(x => Enumerable.SequenceEqual(x.Date.Split(new char[] { '.', '/' }), i.ToShortDateString().Split(new char[] { '.',

        if (infoDay.Count != 0)
        {
            int count = 0;
            string timeWork = infoDay[0].Time;
            int timeWork_ = 0;

            string[] StartTime = db.Staffs.Where(x => x.ID_staff == idStaff).SingleOrDefault().Start_work.Split(new char[] { '.', '/', ':' });
            string[] EndTimeWork = db.Staffs.Where(x => x.ID_staff == idStaff).SingleOrDefault().End_work.Split(new char[] { '.', '/', ':' });
            string[] NowTime = infoDay[0].Time.Split(new char[] { ':', '.' });

            totalWork += Convert.ToInt32(EndTimeWork[0]) * 60 + Convert.ToInt32(EndTimeWork[1]) -
                Convert.ToInt32(StartTime[0]) * 60 - Convert.ToInt32(StartTime[1]);

            if (Convert.ToInt32(StartTime[0]) <= Convert.ToInt32(NowTime[0]) && Convert.ToInt32(StartTime[1]) < Convert.ToInt32(NowTime[1]))
            {
                numberDelays++;
            }

            for (int j = 1; j < infoDay.Count(); ++j)
            {
                if (infoDay[j].check == "free")
                {
                    count++;

                    string[] a = timeWork.Split(new char[] { ':', '.' });
                    string[] b = infoDay[j].Time.Split(new char[] { ':', '.' });
```

Рисунок 5.13 – Реалізація виводу інформації для бухгалтера

В даному коді формуються всі необхідні дані для створення графіків у вікні бухгалтера.На рисунку 5.14 зображено реалізацію обрахунку ефективності персоналу.



```
VSecurity.cs [Конструктор]   Entrance_to_room.cs   CSecurity.cs   VAccountant.cs*   CSign_up.cs   App.config   CAccountant.cs
WEB   WEB.View.VAccountant   button1_Click_1(object sender, EventArgs e)

154
155                 case "Non-working hours":
156                     Info2(asd);
157                     break;
158
159             }
160
161
162             label1.Text = "Number of delays: " + asd.numberDelays;
163             label2.Text = "Working rate: " + Convert.ToInt32(asd.TotalWork) / 60;
164             label4.Text = "Hours worked: " + Convert.ToInt32(asd.HoursWorked) / 60 + ":" + Convert.ToInt32(asd.HoursWorked) % 60;
165
166             double Efficiency = Math.Round(Convert.ToDouble(asd.HoursWorked) / Convert.ToInt32(asd.TotalWork) * 100, 2);
167             if (Efficiency < 75)
168             {
169                 label5.ForeColor = Color.Red;
170             }
171             else
172             {
173                 if (Efficiency < 85)
174                 {
175                     label5.ForeColor = Color.Orange;
176                 }
177                 else
178                 {
179                     if (Efficiency < 95)
180                     {
181                         label5.ForeColor = Color.LightGreen;
182                     }
183                     else
184                     {
185                         label5.ForeColor = Color.DarkGreen;
186                     }
187                 }
188             }
189
190             if (Efficiency > 100)
191             {
192                 label5.Text = "100+ %";
193             }
194             else
195             {
196                 label5.Text = Efficiency.ToString() + " %";
197             }
198         }
199
200         private void Info1(InfoWorkStaff item)
```

Рисунок 5.14 – Реалізація виводу обрахунку ефективності персоналу

Efficiency – це змінна яка містить відсоток ефективності. Обраховується відсоток по такій формулі:

$$\text{Efficiency} = \frac{\text{відпрацьований час}}{\text{робочий час}} * 100,$$

де робочий час – кількість годин, які оюдина повинна відпрацювати за певний проміжок часу- тиждень, місяць і так далі.Відпрацьований час – кількість годин відпрацьованих за певний проміжок часу[17].

Це дозволяє проводити порівняння між всім робочим персоналом і оцінювати їхню ефективність роботи протягом вибраного періоду.

На рисунку 5.15 зображено реалізацію для побудови графіків у вікні бухгалтера

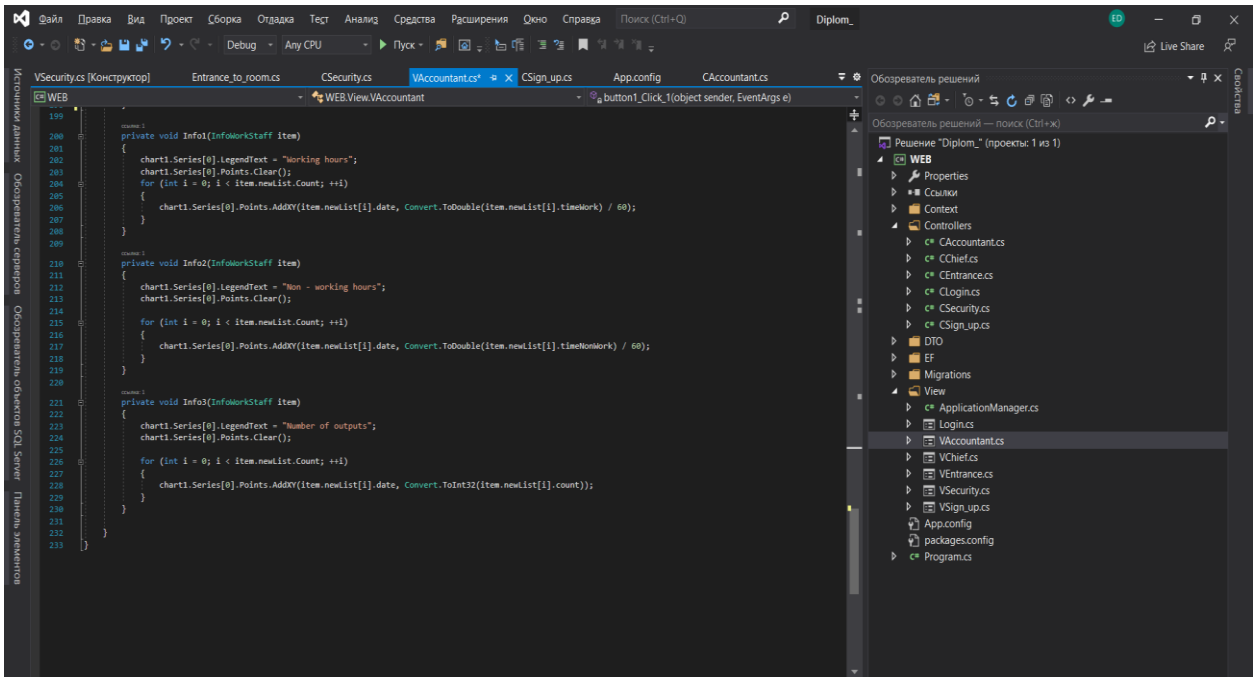


Рисунок 5.15 – Реалізація графіків для бухгалтера

В залежності який тип графіку буде вибраний у самому вікні, такий буде визиватись метод. оскільки використовується один елемент chart (стандартний елемент windows form для побудови графіків), тому одночасно вивести декілька графіків неможливо.

5.3 Створення представлення

Самим представленням виступає WinForms. На рисунку 5.16 зображено головну форму керівника.

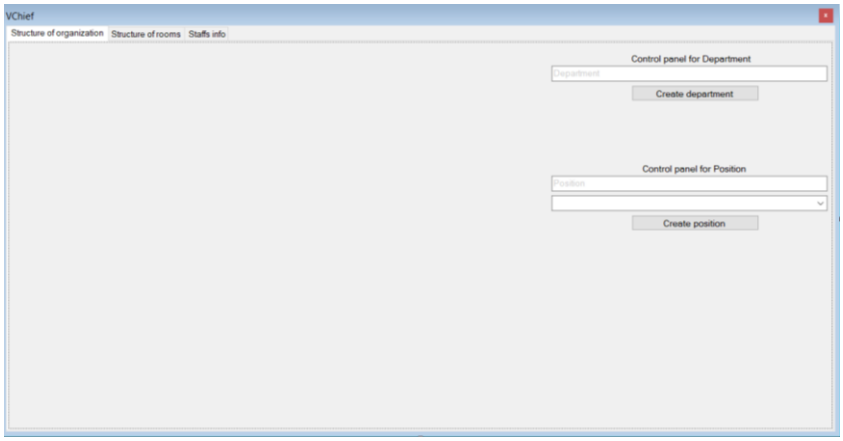


Рисунок 5.16 – форма керівника

За допомогою засобів VisualStudio, ми графічно можемо добавляти нові елементи на представлення. Самі елементи є стандартними засобами Visual Studio. На рисунку 5.17 зображено доступні елементи для роботи з формами.

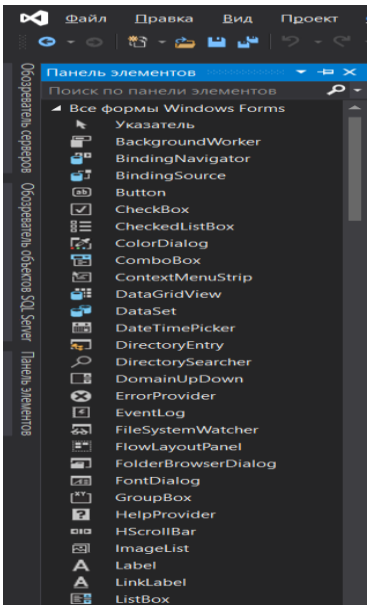


Рисунок 5.17 – елементи для роботи з формами

У представленні створенний додатковий клас, який зображено на рисунку 5.18.

```
public class ApplicationManager
{
    private static ApplicationContext _context;
    Ссылок: 4
    public static ApplicationContext Context
    {
        get
        {
            if (_context == null)
                _context = new ApplicationContext();

            return _context;
        }
    }

    Ссылок: 5
    public static void ShowForm(Form form)
    {
        Form prev = Context.MainForm;
        Context.MainForm = form;

        prev.Close();
        form.Show();
    }
}
```

Рисунок 5.18 – Реалізація класу

Клас використовується для зміни форми. Оскільки у проекті WinForms, є поняття головної форми, то при її закритті, закривається вся програма.Оскільки є різні представлення для ролей, це потрібно було виправити. Метод ShowForm міняє головну форму, на ту яку приймає в параметрах. На рисунку 5.19 зображено обробку подій керівника.

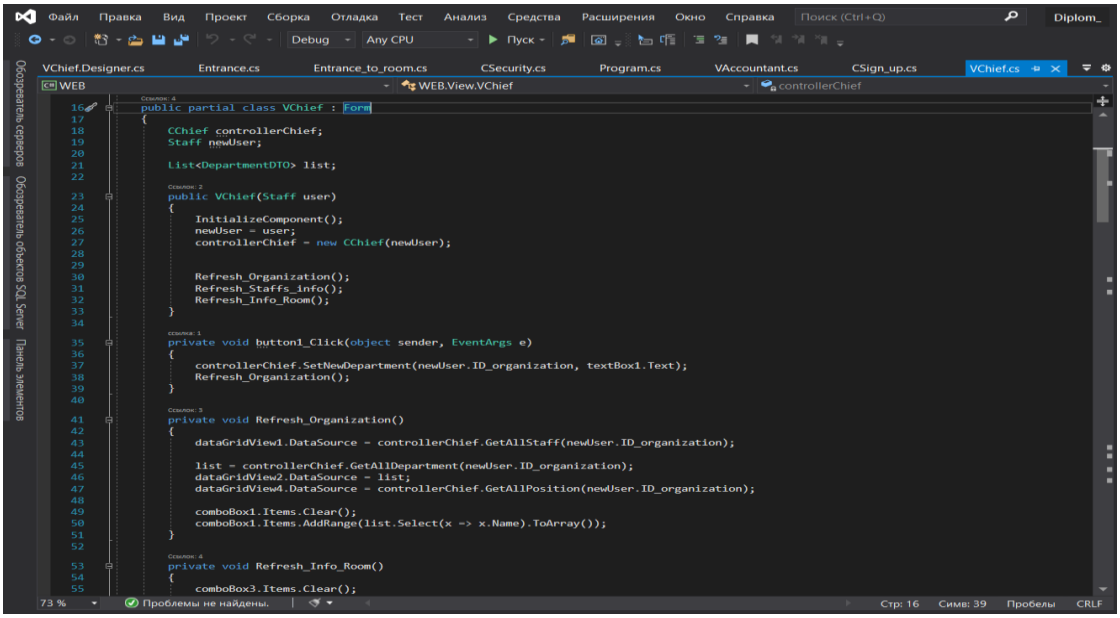


Рисунок 5.19 – Обробник подій для форми керівника

Обробник подій використовується для різних елементів форми. При натисканні на кнопку: створити нового користувача, буде визиватись код прописаний саме в обробнику подій. На рисунку 5.20 зображено обробку подій кнопки форми реєстрації.

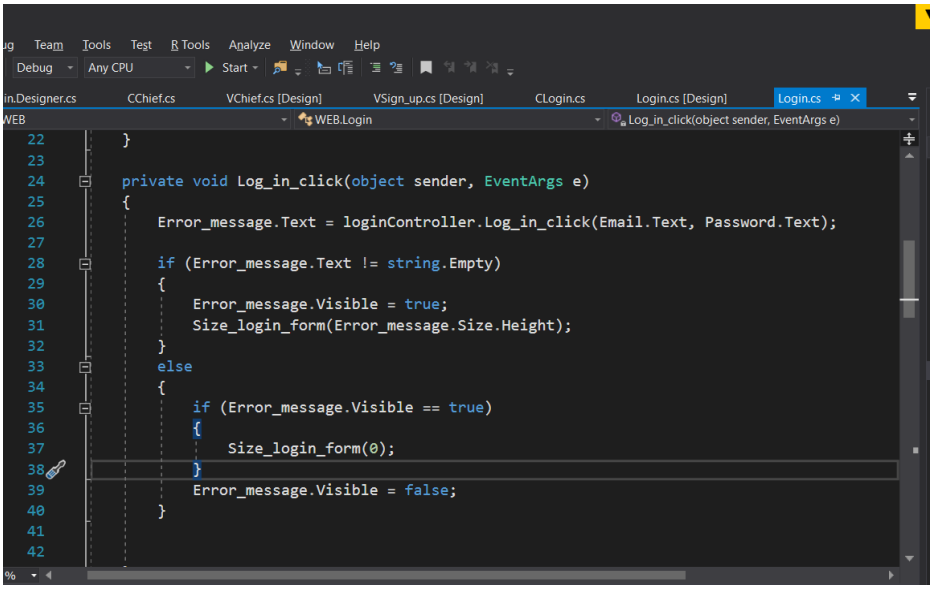


Рисунок 5.19 – Обробник подій кнопки форми реєстрації.

6 РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для забезпечення безвідмовної роботи системи треба дотримуватися основних вимог при інсталяції та рекомендацій щодо її використання.

6.1. Системні вимоги та інсталяція

Для встановлення розробленої програмної системи персональний комп'ютер повинен мати процесор Intel ® Core ™ 2 / 2 Quad / Pentium ® / Celeron ® / Xeon™ чи AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™з тактовою частотою не нижче 2 GHz, на комп'ютері повинна бути встановлена операційна система Windows 10, Windows 8, Windows 7. Крім того, на персональному комп'ютері повинні бути встановлені MySql Service, Microsoft .NET Framework 4.

Для запуску та експлуатації розробленої системи інсталяція не потрібна, достатньо запустити виконуваний файл Start.exe, після цього система запуститься і з'явиться вікно програми.

6.2. Сценарій роботи користувача з системою

Користувач взаємодіє з графічним інтерфейсом, що доступний як десктопний додаток. Оскільки авторизація при запуску програми відбувається, то користувач одразу бачить форму для входу в систему, зображено на рисунку 6.1.

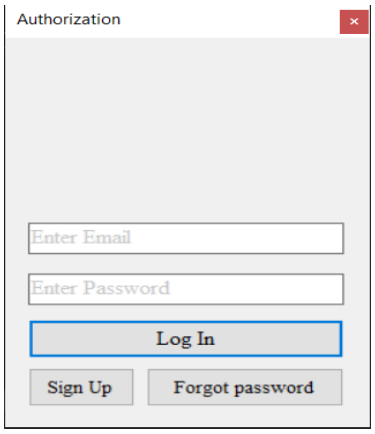
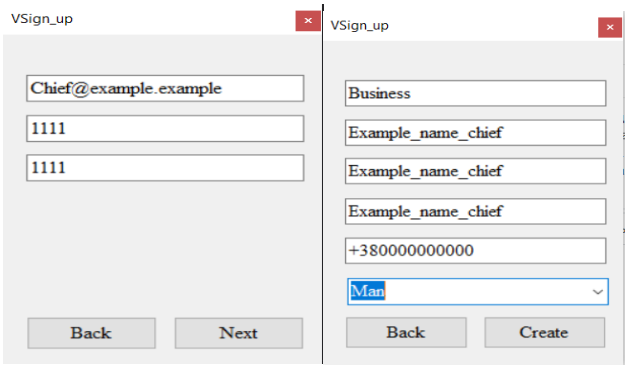


Рисунок 6.1 – Авторизація в системі

В системі використовуються різні ролі: роль керівника,бухгалтера, системного адміністратора або охоронця в залежності хто авторизується, відобразиться представлення для тієї ролі.Якщо користувач, немає акаунту він може тільки зареєструватися як керівник.

Інформацію про працівників інших посад, професій можуть бути доданими до бази даних, але не мають доступу до програмного запезпечення.

На рисунках 6.2-6.3 зображено форми для створення нового акаунту.



Рисунки 6.2, 6.3 – Реєстрація в системі як керівник

Після заповнення даних, нажимаємо create, система створить новий акаунт керівника і відразу перейде на його представлення.

На рисунку 6.4 зображено робочий інтерфейс керівника.

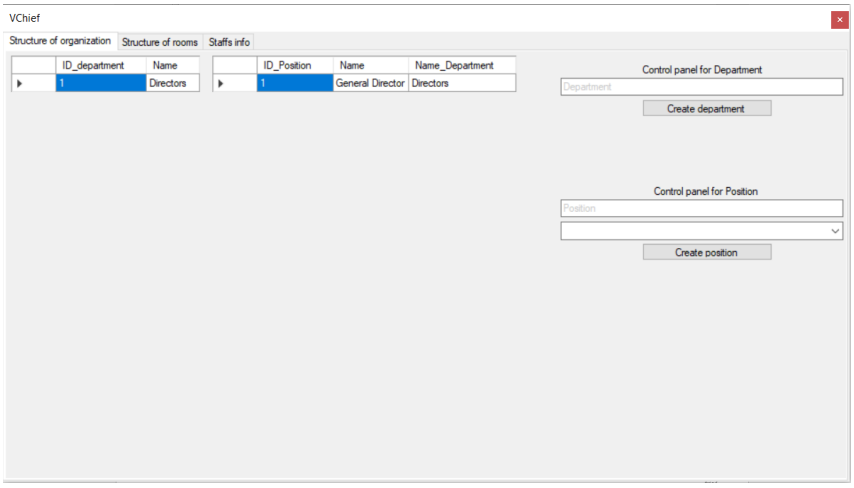


Рисунок 6.4 – Робочий інтерфейс керівника

В першій вкладці інтерфейсу, керівник створює структуру своєї фірми/навчального закладу, є можливість добавляти нові відділи і посади в цих відділах.

При заповненні даних в посаді, ми можемо вибрати відділ, який уже було додано/створено з раніше.

На рисунках 6.5-6.6 зображено введення даних про ввіділ/посаду в БД.

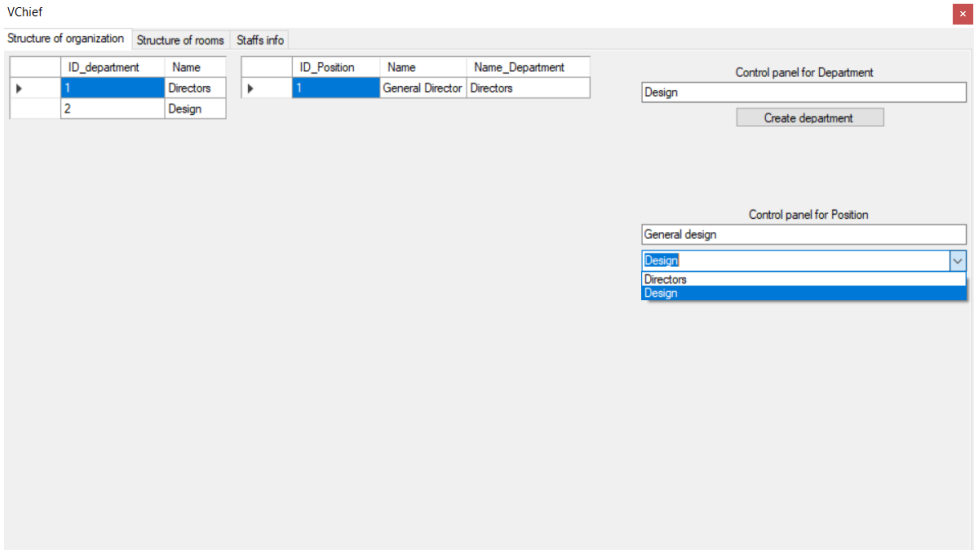


Рисунок 6.5 – Приклад введення даних про ввіділ/посаду в БД

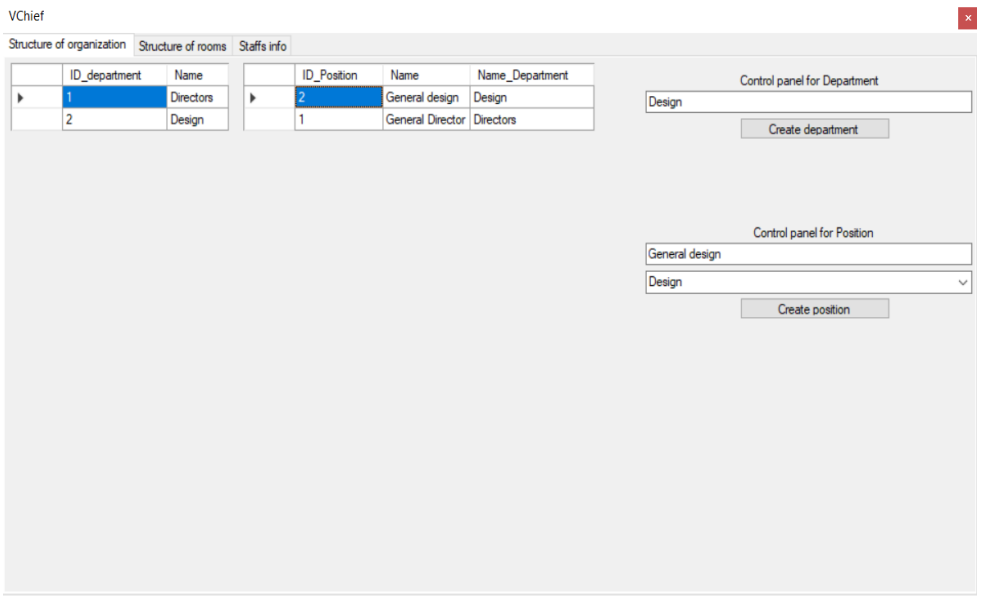


Рисунок 6.6 – Приклад введення даних про відділ/посаду в БД

Друга вкладка відповідає, за заповення структури приміщень організації з рівнями доступів до цих приміщень, зображено на рисунку 6.7.

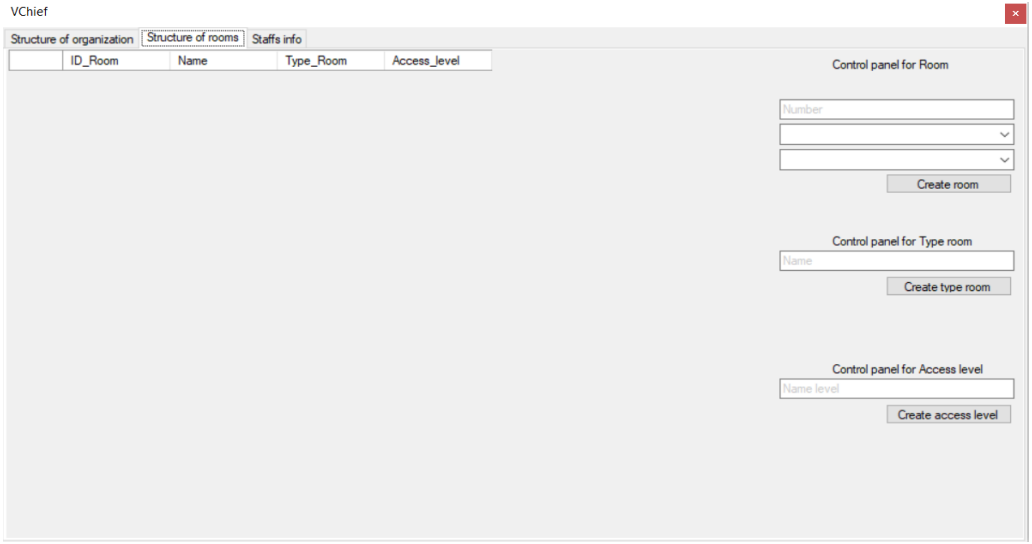


Рисунок 6.7 – Приклад введення даних структури приміщень в БД

На рисунку 6.8 зображено Приклад введення даних структури приміщень в БД

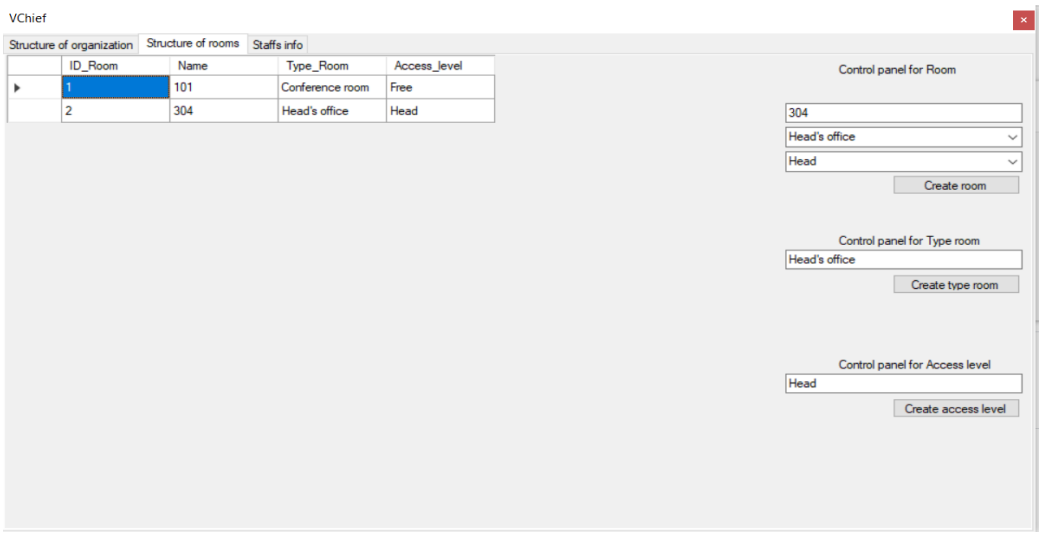


Рисунок 6.8 – Приклад введення даних структури приміщень в БД

Третя вкладка відповідає за персонал організації,в яку заповнюється інформація про працівників, зображено на рисунку 6.9.

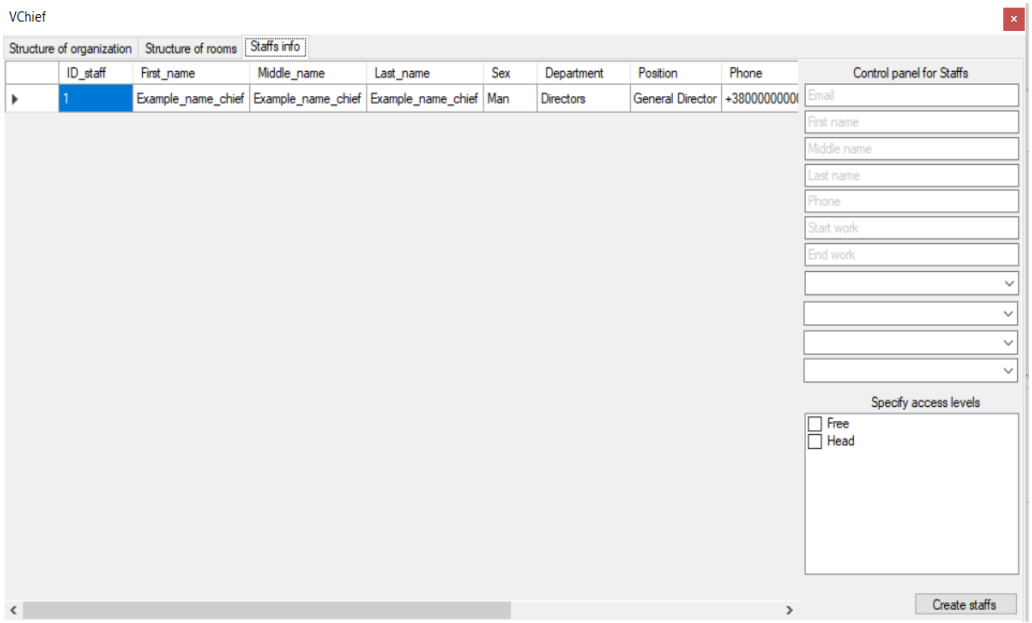


Рисунок 6.9 – Вкладка персоналу

Коли ми добавляємо нового користувача ми вказуємо до яких рівнів системи безпеки він/вона матиме доступ. На рисунку 6.10 зображено інформацію про персонал.

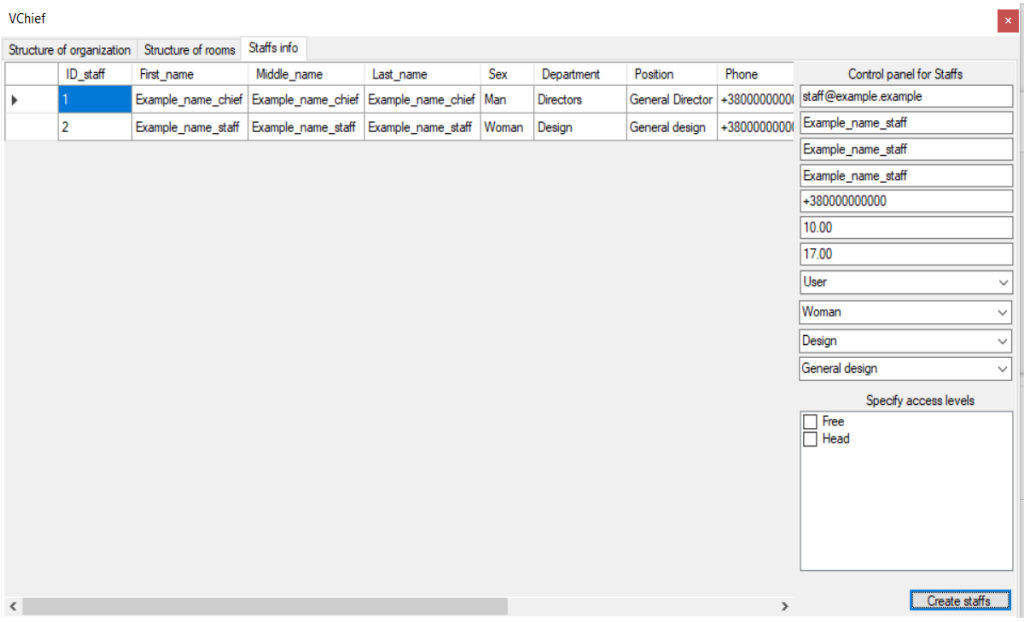


Рисунок 6.10 – Вкладка персоналу з заповненою інформацією

При створенні нових користувачів, для них генерується пароль автоматично в цілях безпеки та для зручності роботи з програмою, зображено на рисунку 6.11.

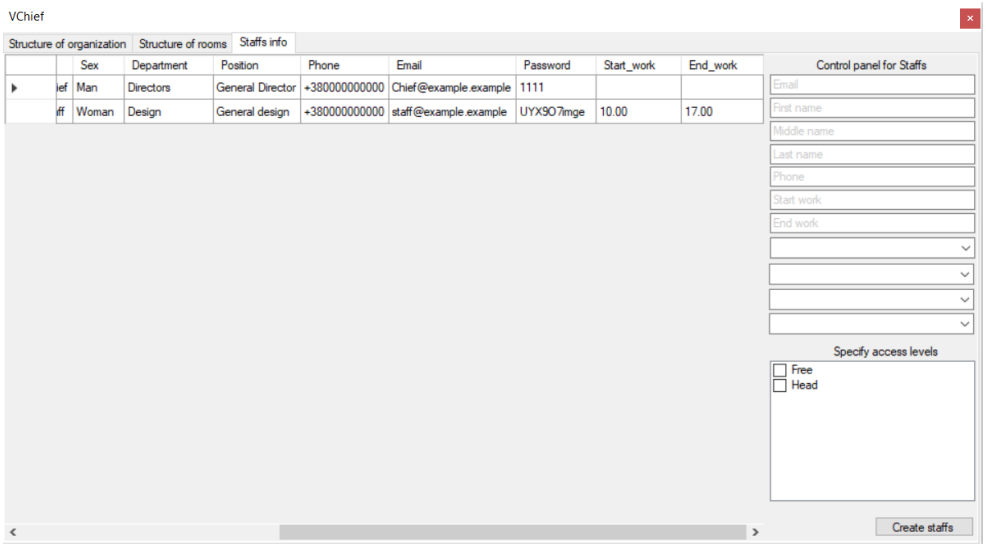


Рисунок 6.11 – Вкладка персоналу з генеруваням автоматичного паролю

Всі дані крім id, ми скрізь можемо редагувати за допомогою елемента winForm: dataGridView в базі даних. На рисунку 6.12 зображено зміну імені в другого користувача.

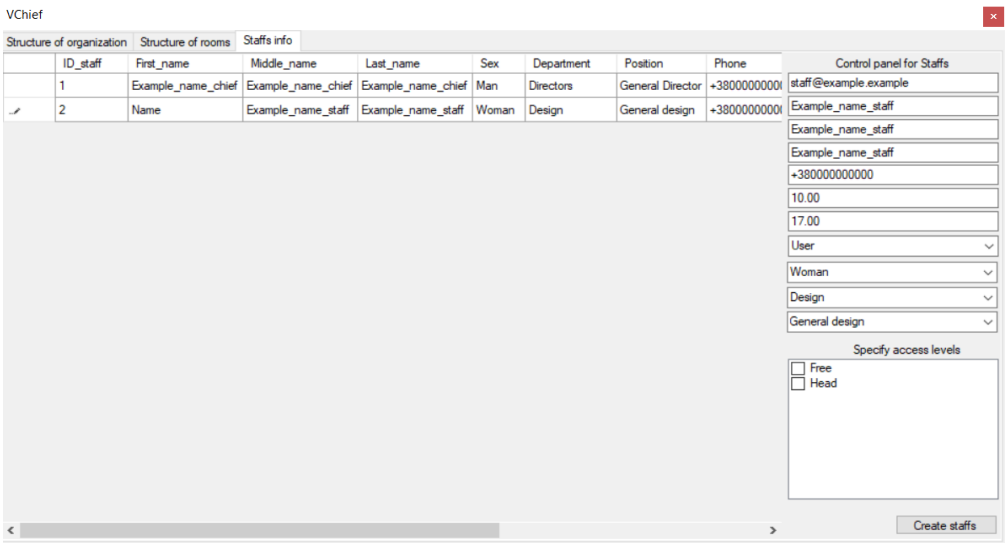


Рисунок 6.12 – Редагування даних імені користувача

При закритті форми у нас відкривається заново вікно входу, зображено на рисунку 6.13.

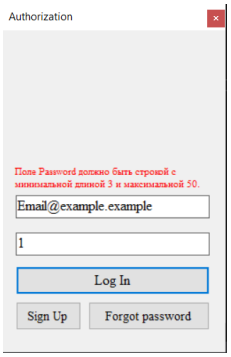


Рисунок 6.13 – Приклад валідації даних

Якщо авторизуватись як бухгалтер, то буде інша форма. Перша вкладка, як і в керівника дозволяє вносити дані про інший персонал.На другій вкладці можна знаходити інформацію,зокрема аналітичні дані про персонал, за допомогою прізвища та ім’я протягом останнього місяця. Її середній час роботи, кількість виходів, порушень, кількість запізнень. На рисунках 6.14-6.15 зображено два робочих інтерфейси бухгалтера.

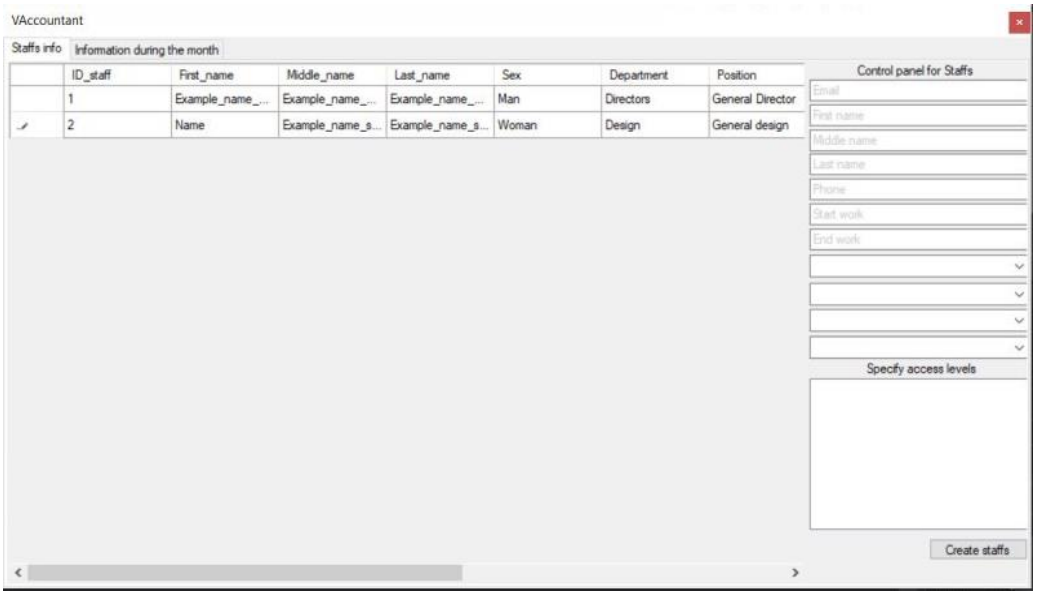


Рисунок 6.14 – Вкладка персоналу для бухгалтера для пошуку інформації

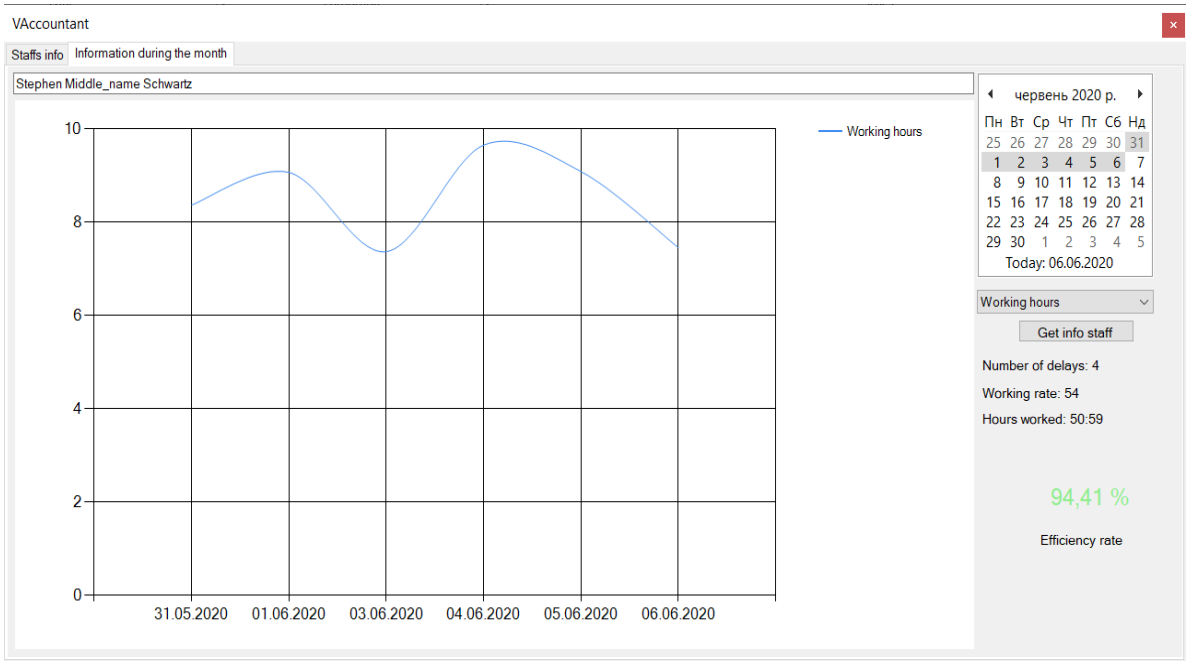


Рисунок 6.15 – Графік відпрацьованих годин

Для відображення інформації потрібно ввести ПІП користувача якого хочемо знайти. Потрібно вибрати період відображення в календарі і тип графіку для відображеняя.

На графіку зображено інформацію про робочий час користувача протягом певного періоду, також відображається інформація про кількість його пропусків, загальний час який він мав відпрацювати і відпрацьований час. Знаходиться його ефективність роботи на робочому місці порівнючи відпрацьовані години до норми годин , які повинні бути відпрацьованними, і відображається за допомогою відсотків.

На рисунку 6.16 зображено невідпрацьовнні години протягом вибраного періоду.Невідпрацьовані години сумуються за один робочий день та за вибраний користувачем період.

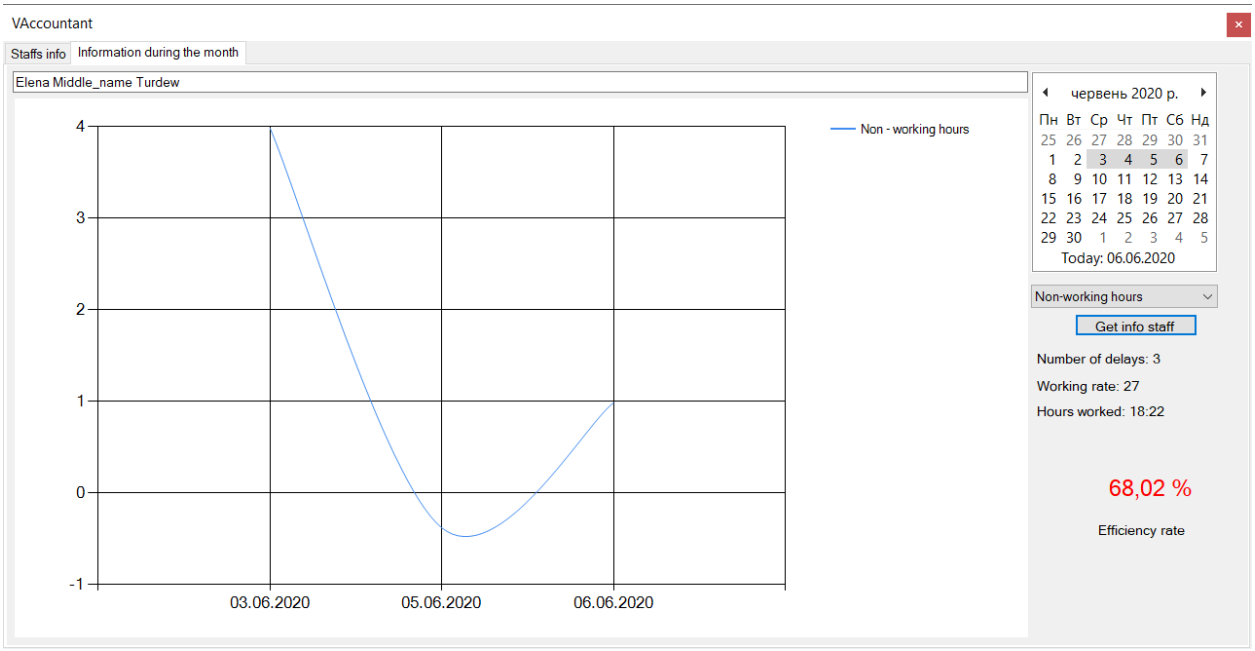


Рисунок 6.16 – Графік невідпрацьованих годин вибраного користувача

На рисунку 6.17 зображено графік кількість виходів протягом вибраного періоду.

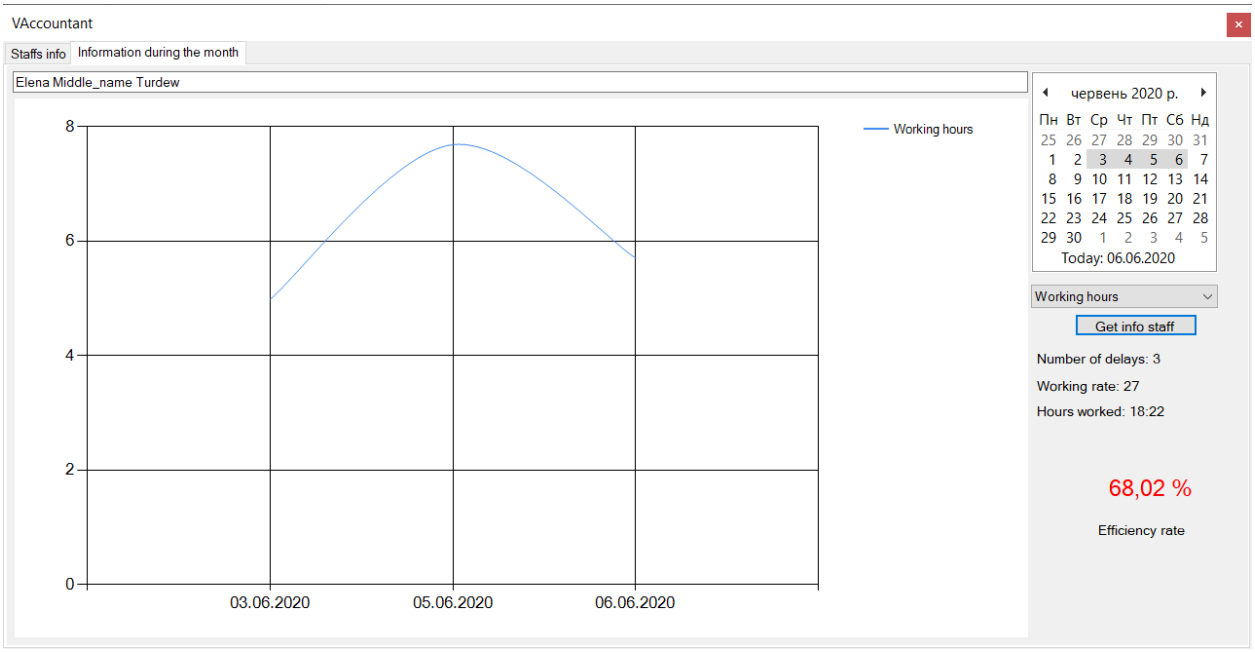


Рисунок 6.17 – Інформація про кількість виходів

На рисунку 6.18 зображено головне вікно охорони.

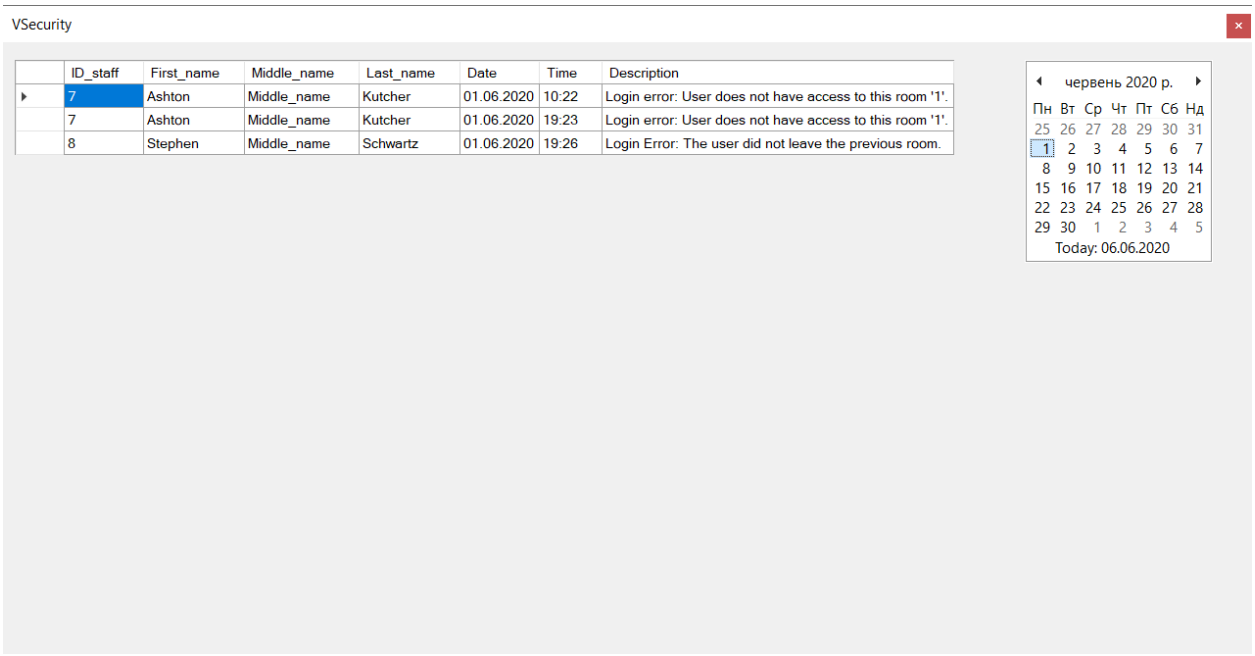


Рисунок 6.18 – Головне вікно охорони

Тут відображаються всі помилки зчитування про вхід/вихід через пропускну систему протягом вибраного періоду.Наступні дані про помилки відображаються у

вікні охорони:випадки проходу декілька разів по одній картці без виходу через пропускну систему, спроба пройти до частини будівлі до якої немає доступу або прохід через систему в неробочий час. Інформацію можна дивитись, як окремо по днях так і відразу протягом місяця.

На рисунку 6.19 зображено форму для внесення даних про вхід користувача

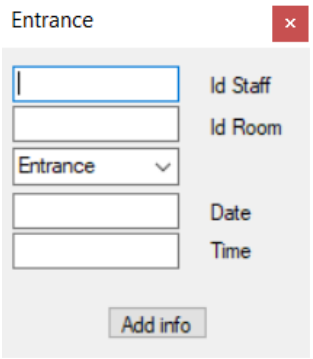


Рисунок 6.19 – Форма для заповнення даних про вхід/вихід

Дана форма використовується замість самого механізму проходу в приміщення.

В даному розділі були описані системні вимоги персонального комп`ютера для інсталяцію та запуску програми.Сценарій роботи користувача з системою був доданий.Інтерфейс програми для різних користувачей був описаний, цілі різних частин програми були обгрунтовані.

ВИСНОВКИ

Отже, під час виконання дипломної роботи було проведено дослідження предметної області та розроблено програмне забезпечення з базовими статистичними показниками аналітики даних. Було досліджено та проаналізовано існуючі програмні рішення, які водночас є доступними комерційними продуктами.Було проведено порівняння між цими програмними рішеннями та дипломною роботою. Обгрунтовано актуальність створення нового продукту як для аналітики ефективності праці, так і для аналітики системи контролю та доступу. Користувачі системи та середовище застосування були визначиними. Були розроблені різні режими програми для різних цілей:аналітика ситеми контролю та захисту, аналітика продуктивності працівників.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Unstructured Data Analytics: How to Improve Customer Acquisition, Customer Retention, and Fraud Detection and Prevention – by Jean Paul Isson – 26ст.

2. The benefits of Big Data Analytics in security – Dr. Vibhor Gupta, Ph.D., Technology Lead, ASIS UK [Електронний ресурс] – Режим доступу до ресурсу: https://citysecuritymagazine.com/security-technology/big-data-analytics-in-security/

3. India companies use analytics tools to spot talent and trouble [Електронний ресурс] – Режим доступу до ресурсу:https://timesofindia.indiatimes.com/trend-tracking/India-companies-use-analytics-tools-to-spot-talent-and-trouble/articleshow/54790024.cms

4. An Introduction to Statistical Learning: With Applications in R – Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani-20ст.

5. Electronic Access Control – by Thomas L. Norman

6. Statistics [Електронний ресурс] - Режим доступу до ресурсу: https://www.investopedia.com/terms/s/statistics.asp

7. Programming Entity Framework: Code First - Book by Julia Lerman and Rowan Miller– 54ст.

8. LINQ to Entities [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/linq-to-entities

9. Microsoft .NET Framework 4.8 [Електронний ресурс] – Режим доступу до ресурсу: https://www.techspot.com/downloads/4297-microsoft-net-framework.html

10. What is SQL Management Studio(SSMS) [Електронний ресурс] – Режим доступу до ресурсу:https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms#:~:text=SQL%20Server%20Management%20Studio%20(SSMS)%20is%20an%20integrated%20environment%20for,Database%2C%20and%20SQL%20Data%20Warehouse.

11. MVC Design Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/mvc-design-

pattern/#:~:text=The%20Model%20View%20Controller%20(MVC,but%20not%20for%20complete%20application.

12. Pro ASP.NET MVC 5 by Adam Freeman – 125ст.

13. Map Many – to Many Relationships [Електронний ресурс] – Режим доступу до ресурсу:https://docs.microsoft.com/en-us/sql/ssms/visual-db-tools/map-many-to-many-relationships-visual-database-tools?view=sql-server-ver15

14. Types of Entities in Entity Framework [Електронний ресурс] – Режим доступу до ресурсу:https://www.entityframeworktutorial.net/Types-of-Entities.aspx#:~:text=A%20POCO%20entity%20is%20a,EF%206%20and%20EF%20Core.

15. Understanding Model,View and Controllers(C#) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/overview/understanding-models-views-and-controllers-cs

16. Create Data Transfer Objects(DTOs) [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5

17. Big Data Analytics and Organizational Performance: A Meta-Analysis Study – Mihai Bogdan – 231ст.

ДОДАТОК А

Модуль розробки аналітики даних для систем контролю та безпеки

Специфікація

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ТР62140_20Б

Аркушів 2

Київ – 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТР62140_20Б 81-1	Пояснювальна записка.docx	Пояснювальна записка
Комплекс		
Компоненти		
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТР62140_20Б 13-1	Додаток В	Опис програмного коду
УКР.НТУУ «КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТР62140_20Б 12-1	Projekt.sln	Основний компонент, що включає в себе інші

ДОДАТОК Б

Модуль розробки аналітики даних для систем контролю та безпеки

Лістинг програми

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТР62140_20Б

Аркушів 9

Київ – 2020

```
public class CChief
{
    Staff newUser;

    public CChief(Staff newuser)
    {
        newUser = newuser;
    }

    public List<StaffDTO> GetAllStaff(int id)
    {
        using (UserContext db = new UserContext())
        {
            List<Staff> rtn = db.Staffs.Where(x => x.ID_organization == id).ToList();

            for (int i = 0; i < rtn.Count; ++i)
            {
                rtn[i].Sex = db.Sexes.Find(rtn[i].ID_sex);
                rtn[i].Position = db.Positions.Find(rtn[i].ID_position);
                rtn[i].Position.Department = db.Departments.Find(rtn[i].Position.ID_department);
            }
            List<StaffDTO> list_ = new List<StaffDTO>();
            for (int i = 0; i < rtn.Count; ++i)
            {
                list_.Add(new StaffDTO
                {
                    ID_staff = rtn[i].ID_staff,
                    First_name = rtn[i].First_name,
                    Middle_name = rtn[i].Middle_name,
                    Last_name = rtn[i].Last_name,
                    Sex = rtn[i].Sex.Name,
                    Department = rtn[i].Position.Department.Name,
                    Position = rtn[i].Position.Name,
                    Phone = rtn[i].Phone,
                    Start_work = rtn[i].Start_work,
                    End_work = rtn[i].End_work,
                    Email = rtn[i].Email,
                    Password = rtn[i].Password,
                });
```

```
        }
        return list_;
    }
}

public List<PositionDTO> GetAllPosition(int id)
{
    List<PositionDTO> list = new List<PositionDTO>();
    using (UserContext db = new UserContext())
    {
        List<Position> list_;
        IQueryable<Position> rtn = from temp in db.Positions select temp;
        list_ = rtn.ToList();

        for (int i = 0; i < list_.Count; ++i)
        {
            list_[i].Department = db.Departments.Find(list_[i].ID_department);
        }

        list_ = list_.Where(x => x.Department.ID_organization == id).ToList();

        for (int i = 0; i < list_.Count; ++i)
        {
            list.Add(new PositionDTO { ID_Position = list_[i].ID_position, Name = list_[i].Name, Name_Department =
                    list_[i].Department.Name });
        }

        list = list.OrderBy(x => x.Name_Department).ToList();
        return list;
    }
}

public List<DepartmentDTO> GetAllDepartment(int id)
{
    List<DepartmentDTO> list = new List<DepartmentDTO>();

    using (UserContext db = new UserContext())
    {
        IQueryable<Department> rtn = from temp in db.Departments.Where(x => x.ID_organization == id) select temp;
```

```
        List<Department> list_ = rtn.ToList();

        for (int i = 0; i < list_.Count; i++)
        {
            list.Add(new DepartmentDTO { Name = list_[i].Name, ID_department = list_[i].ID_department });
        }
        return list;
    }
}

public void SetNewDepartment(int id, string name)
{
    using (UserContext db = new UserContext())
    {
        Department newDepartment = new Department { Name = name, ID_organization = id };
        db.Departments.Add(newDepartment);
        db.SaveChanges();
    }
}

public void SetNewPosition(int id, string name)
{
    using (UserContext db = new UserContext())
    {
        Position newPosition = new Position { Name = name, ID_department = id };
        db.Positions.Add(newPosition);
        db.SaveChanges();
    }
}

public List<Role> GetRole()
{
    using (UserContext db = new UserContext())
    {
        IQueryable<Role> rtn = from temp in db.Role select temp;
        return rtn.ToList();
    }
}
```

```
public Role GetRole(string name)
{
   using (UserContext db = new UserContext())
   {
      return db.Role.Where(x => x.Name == name).FirstOrDefault();
   }
}

public List<Sex> GetSex()
{
   using (UserContext db = new UserContext())
   {
      IQueryable<Sex> rtn = from temp in db.Sexes select temp;
      return rtn.ToList();
   }
}

public Sex GetSex(string name)
{
   using (UserContext db = new UserContext())
   {
      return db.Sexes.Where(x => x.Name == name).FirstOrDefault();
   }
}

public List<Department> GetDepartment()
{
   using (UserContext db = new UserContext())
   {
      IQueryable<Department> rtn = from temp in db.Departments.Where(x => x.ID_organization ==
            newUser.ID_organization) select temp;
      return rtn.ToList();
   }
}

public Department GetDepartment(string name)
{
   using (UserContext db = new UserContext())
   {
```

```
        return db.Departments.Where(x => x.Name == name).FirstOrDefault();
    }
}

public List<Position> GetPosition(string nameDepartment)
{
    using (UserContext db = new UserContext())
    {
        var rtn = db.Departments.Where(x => x.Name == nameDepartment && x.ID_organization ==
                newUser.ID_organization).SingleOrDefault();
        List<Position> list_;
        IQueryable<Position> rtn_ = from temp in db.Positions.Where(x => x.ID_department == rtn.ID_department) select temp;
        list_ = rtn_.ToList();

        return list_;
    }
}

public Position GetPosition_(string name)
{
    using (UserContext db = new UserContext())
    {
        return db.Positions.Where(x => x.Name == name).FirstOrDefault();
    }
}

public void SetNewStaff(Staff newStaff)
{
    using (UserContext db = new UserContext())
    {
        List<Staff> check = db.Staffs.Where(x => x.Email == newStaff.Email).ToList();
        if (check.Count != 0)
        {
            MessageBox.Show("Email already exists", "Error", MessageBoxButtons.OK);
        }
        else
        {

            db.Staffs.Add(newStaff);
```

```
            db.SaveChanges();
        }
    }
}

public List<RoomDTO> GetAllRoom(int id)
{
    using (UserContext db = new UserContext())
    {
        List<Room> list__ = db.Rooms.ToList();

        for (int i = 0; i < list__.Count; ++i)
        {
            list__[i].Room_type = db.Room_types.Find(list__[i].ID_Room_type);
            list__[i].Access_level = db.Access_Levels.Find(list__[i].ID_Access_level);
        }

        List<RoomDTO> newList = new List<RoomDTO>();
        list__ = list__.Where(x => x.Room_type.ID_Organization == id).ToList();

        for (int i = 0; i < list__.Count; ++i)
        {
            newList.Add(new RoomDTO
            {
                ID_Room = list__[i].ID_Room,
                Name = list__[i].Name,
                Type_Room = list__[i].Room_type.Name,
                Access_level = list__[i].Access_level.Name_level
            });
        }

        return newList;
    }
}

public List<Room_type> GetRoom_type()
{
    using (UserContext db = new UserContext())
    {
```

```
        IQueryable<Room_type> rtn = from temp in db.Room_types.Where(x => x.ID_Organization ==
                newUser.ID_organization) select temp;
        return rtn.ToList();
    }
}

public Room_type GetRoom_type(string name)
{
    using (UserContext db = new UserContext())
    {
        return db.Room_types.Where(x => x.Name == name).FirstOrDefault();
    }
}

public void CreateNewTypeRoom(string name)
{
    using (UserContext db = new UserContext())
    {
        db.Room_types.Add(new Room_type { Name = name, ID_Organization = newUser.ID_organization });
        db.SaveChanges();
    }
}

public void CreateNewRoom(string name, string type, string access)
{
    using (UserContext db = new UserContext())
    {
        db.Rooms.Add(new Room { Name = name, ID_Room_type = db.Room_types.Where(x => x.Name ==
                type).SingleOrDefault().ID_Room_type,
         ID_Access_level = db.Access_Levels.Where(x => x.Name_level == access).SingleOrDefault().ID_Access_level });
        db.SaveChanges();
    }
}

public void CreateNewAccesslevel(string nameLevel)
{
    using (UserContext db = new UserContext())
    {
        db.Access_Levels.Add(new Access_level { Name_level = nameLevel });
```

```
            db.SaveChanges();
        }
    }

    public List<Access_level> GetAccess_level()
    {
        using (UserContext db = new UserContext())
        {
            IQueryable<Access_level> rtn = from temp in db.Access_Levels select temp;
            return rtn.ToList();
        }
    }

    public Staff GetOneStaff(string email)
    {
        using (UserContext db = new UserContext())
        {
            return db.Staffs.Where(x => x.Email == email).SingleOrDefault();
        }
    }

    public Access_level GetOneAccess(string name)
    {
        using (UserContext db = new UserContext())
        {
            return db.Access_Levels.Where(x => x.Name_level == name).SingleOrDefault();
        }
    }

    public void CreateNewAdditional(Additional_access_staffs newItem)
    {
        using (UserContext db = new UserContext())
        {
            db.Additional_Access_Staffs.Add(newItem);
            db.SaveChanges();
        }
    }
}
```

ДОДАТОК В

Модуль розробки аналітики даних для систем контролю та безпеки

Опис програмного модулю

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_ТЕФ_АПЕПС_ ТР62140_20Б

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Додаток являє собою систему аналітики для систем контролю та доступу.У функціонал входить: можливість створення всієї необхідної інформації про користувачів системи,рівні доступу, додання інформації про прохід через систему, перегляд всієї інформації у формі таблиць та графіків.

Модуль було розроблено з використанням мови програмування C#, фреймворку Entity Framework, технології MVC, Ms Sql, середовища розробки Visual Studio Code.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

Дану програмну систему було розроблено з використанням середовища Visual Studio 2019 мовою програмування C# з використанням Windows Form та технологій MVC, Entity Framework.

Програма призначена відображення аналітики даних створеною на основі інформації про проходи через фізичну пропускну систему.

До основних переваг програми можна віднести швидкість роботи та відносно простий для користувача інтерфейс.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

При створенні програмного продукту була поставлена задача створення системи аналітики даних для системи аналітики контролю та захисту. Для зберігання даних була створена база даних. За допомогою технології Entity, базу даних можна легко розширювати і змінювати. Програмний продукт містить декілька різних типів методів, які виконують наступні функції:

- швидка взаємодія з базою даних;
- додавання інформації про користувачів та проходи через систем;
- можливість перегляду всієї інформації;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Створена програма складається з декількох частин:

- бази даних (створення структури бази даних);
- бізнес логіки (проміжний модуль, який працює з базою даних і представленням);
- представлення (робота користувача з представленням у вигляді Windows Form).

Робота програми побудована таким чином, що відкривши Projekt.sln користувач відкриває форму авторизації. Авторизувавшись перед користувачем постає ряд можливостей:

- заповнити всю необхідну інформацію про користувачів та прохід через систему;
- перегляд всієї інформації про помилки проходу через систему;
- перегляд всієї інформації включачи графіки про ефективінсть кожного працівника;

ВИКОРИСТУВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для створення даної програми було використано середовище розробки Visual Studio 2019 та мова програмування C# з використанням Windows Form та технологій MVC, Entity Framework.

Для зберігання великих об’ємів даних була створена база даних Ms Sql.

Для запуску даної програми користувачу необхідно мати комп’ютер з мінімальними характеристиками.

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

– дані про вхід/вихід через фізичну пропускну систему.

Вихідними даними є:

– аналітичні дані про помилки зафіксовані при проході через систему та аналітичні дані та висновки по ефективності праці.